Artificial boundary conditions for simulations of seismic air-gun bubbles

J. R. C. King

THE UNIVERSITY
of EDINBURGH

Thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy
to the
University of Edinburgh --- 2014

Declaration

I declare that this thesis has been composed solely by myself and that it has not been submitted, either in whole or in part, in any previous application for a degree. Except where otherwise acknowledged, the work presented is entirely my own.

J. R. C. King

December 2014

Abstract

Marine seismic exploration is a method employed by the hydrocarbon industry to find geological structures in the sub-surface with the potential to contain trapped hydrocarbons. A source of seismic energy is towed behind a ship. The energy produced by the source propagates as a sound wave through the sea into the sub-surface. Within the sub-surface the energy is reflected, refracted and diffracted. The ship also tows an array of hydrophones behind the seismic source, and these are used to measure the wavefield. If the source signal is known, then the received signal at each hydrophone can be deconvolved for the source signal to obtain the impulse response of the earth between the source and the hydrophone. These impulse responses can highlight some of the structures in the subsurface. Maps of the subsurface built up from these impulse responses are then interpreted to estimate the locations of trapped hydrocarbons.

The most commonly used seismic source is the seismic air gun, which is a canister containing highly compressed air. The air is released into the sea, forming an oscillating bubble. There are two methods used by industry to determine the signal produced by an air gun or air gun array: (1) modelling, and (2) extrapolation from near-field measurements. Traditionally, industry uses the first method. With broader bandwidth data that are being recovered in data processing by removing the sea-surface reflection at the source and receiver (source and receiver ghosts), it has been found that modelling is inferior to extrapolation from near field measurements, although industry has been slow to adopt the second method. Despite this change, modelling remains a valuable tool in the design of air gun arrays, where designs can be optimised by adjusting

parameters of the array and using modelling to determine the wavefield of each variation of the array. The aim of this work is to develop methods which can improve on current air gun bubble modelling.

In this thesis I develop a novel artificial boundary condition for use in finite volume simulations of oscillating bubbles. The purpose of the work is an improvement to the modelling of seismic air gun bubbles. However, the techniques presented in this thesis are not limited to air gun bubbles, but are applicable to any oscillating bubbles, or indeed any fluid dynamics problem which is spherical in nature, close to spherically symmetric, and produces flow speeds of low (< 0.1) Mach number some distance from the region of interest. The boundary condition is based on an existing approximation to the motion around a spherical bubble, which is derived from the asymptotic solution to the motion in the far field. It is applied as follows: (1) use the solution on the domain boundary to calculate the approximate solution external to the domain; (2) use the approximate external solution to calculate spatial derivatives of properties on the domain boundary, due to the external solution, and (3) use the spatial derivatives to describe characteristic waves incoming to the domain. I develop a finite volume scheme in which I apply this boundary condition. I present the results of one- and two-dimensional of simulations using this scheme, and demonstrate the efficacy of this boundary condition.

The boundary condition performs well, allowing finite volume simulations of bubbles to be carried out for long run-times (5×10^5 time steps with a CFL number of 0.8) on highly truncated domains, in which the boundary condition may be applied within 0.1% of the maximum bubble radius. Conservation errors due to the boundary condition are found to be of the order of 0.1% after 10^5 time steps. One- and two-dimensional results show a third-order convergence rate of errors due to the boundary condition as the domain is enlarged. The one- and two-dimensional simulations of air gun bubbles I present are, to my knowledge, the first finite volume simulations of air gun bubbles carried out, and the first air gun bubble simulations in which the contents of the bubble are not considered to be homogeneous.

Two-dimensional results show non-spherical aspects of air gun bubbles, which may be incorporated into models used by industry. The model captures surface instabilities, bubble translation and deformation due to gravity, and the formation of jets due to asymmetries on collapse. The results indicate that bubble surfaces are unstable throughout collapse. These phenomena are shown to increase the damping of bubble oscillations. The results of the two-dimensional air gun modelling highlight the potential value of my artificial boundary condition, and also the aspects of my computational scheme which require improvement. I extend the numerical scheme to include viscous effects, which I show to have limited impact on the signals emitted by air gun bubbles, although the influence of a boundary layer around the bubble is significant, causing an 18% reduction in rise rates. I extend the scheme to include the effects of the sea surface, and present results which show the impact of the reflection from the sea surface (the ghost wave) on the bubble. This extension shows the reflection of the ghost wave off the bubble, which provides a novel explanation of some of the higher frequencies present in measurements. This extension further increases the practical value of my contribution, and further demonstrates the ability of the boundary condition to handle asymmetrical flow features.

Lay Summary

Marine seismic exploration is a method used by the oil industry to search for oil and gas beneath the sea floor. This method is a form of echo sounding. A seismic air gun, which is a canister of highly compressed air, is towed behind a ship, several metres under the surface. The air is released and forms a bubble, which expands and collapses several times. This process is similar to, although quieter than, setting off an explosion underwater. As the air is released, and then each time the bubble collapses, a sound wave is produced. This sound wave propagates out through the sea and into the sea floor. The sound waves are recorded. Provided the sound produced by the air gun is known, the recordings can be processed to create a map of the geology beneath the sea floor. In order to determine the sound produced by the air gun, industry uses computers to model air gun bubbles. The aim of this work is to develop a better air gun bubble model.

When modelling bubbles, one approach is to place a boundary around the region of interest, in order to reduce the time required for the model to run. The model is confined to the region within this boundary. On the boundary an 'artificial boundary condition' is applied, which aims to mimic the behaviour of everything outside the boundary, and pass this information through the boundary for use by the model. Current models are very simple, and do not use this method. However, the advantage of this method is that allows the model to include more interesting effects, such as the changing shape of the bubble. In this thesis I develop a new artificial boundary condition, which allows me to create a more complex air gun bubble model.

My results show that the errors caused by this approach are small, and it can give significant savings in the time required for the simulations to run. My model shows some interesting aspects of air gun bubbles. As bubbles collapse, their surface becomes unstable, and wrinkles develop. The model shows bubbles rising due to buoyancy, and changing shape as they do so. The sound waves produced by the bubble are reflected off the sea surface. The model captures the way the bubble behaves when these echoes reach it. These results may explain some aspects of measurements which have not previously been understood.

Acknowledgements

Firstly, I must express my gratitude to Professor Anton Ziolkowski for his supervision of this Ph.D. project. Secondly, I would like to thank Max Ruffert, for the many interesting and useful discussions, and also for all the green tea. Thirdly, I thank PGS for funding this project and providing the data. Of those in PGS, I specifically thank Gregg Parkes, Daniel Barker and the source modelling team in Oslo.

Finally, I thank Zahra for her encouragement and support throughout.

Contents

List of Tables

List of Figures

Conventions, definitions, units and symbols

Scalar and vector quantities

Vector quantities are written in a bold typeface, for example $\mathbf{F}$, and scalar quantities in normal typeface, for example F.

Space and time

In general let $\mathbf{r} = (x_1, x_2, x_3)$ denote the position vector in $\mathbb{R}^3$, and $\mathbf{e}_i$ the i-th orthonormal basis vector such that

$$\mathbf{r} = \sum_{i=1}^{3} x_i \mathbf{e}_i.$$

The components of a vector field, $\mathbf{F}$ for example, are denoted $\mathbf{F} = (F_{x_1}, F_{x_2}, F_{x_3})$.

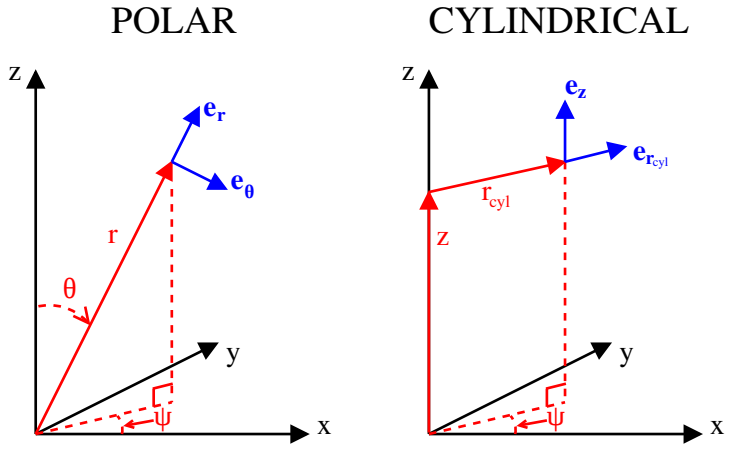


Figure 1: The relations between various coordinate systems.

Whilst the bulk of this thesis uses a polar coordinate system, it is useful to define the coordinate systems used in relation to a right-handed Cartesian frame of reference, as shown in Figure 1. Let $\mathbf{r} = (x, y, z)$ in the Cartesian frame, such that z increases with decreasing depth. In all diagrams, x and z are in the plane of the paper, such that z

increases towards the top of the page and x increases to the right. y increases into the page. For the polar coordinate system, the position vector $\mathbf{r} = (r,\, \theta,\, \psi)$ is defined by

$$r = ||\mathbf{r}||\,, \qquad \cos\theta = z/r \qquad \text{and} \qquad \tan\psi = y/x.$$

For the cylindrical coordinate system, the position vector $\mathbf{r} = (r_{cyl},\, z,\, \psi)$ is defined by

$$r_{cyl} = \left(x^2 + y^2\right)^{\frac{1}{2}}, \qquad z \to z \qquad \text{and} \qquad \tan\psi = y/x.$$

The work in this thesis is limited to problems with spherical or cylindrical symmetry. All variation and motion in the ψ direction is assumed to be zero. Under the assumption of polar axisymmetry, $\mathbf{r} = (x, z)$ in Cartesian coordinates, $\mathbf{r} = (r_{\text{cyl}}, z)$ in cylindrical coordinates and $\mathbf{r} = (r, \theta)$ in polar coordinates. Let t denote time.

Differential operators

Where F is a function of only a single variable, say τ, let F' denote the derivative of F with respect to τ. The exception to this rule is where F is a function of time only, in which case let $\dot{F}$ denote the time-derivative of F. For multivariate functions, $F = F\left(a_1, a_2, a_3 \dots\right)$, let the partial derivative of F with respect to a_n be denoted by

$$\frac{\partial F}{\partial a_n}.$$

In Cartesian coordinates

$$\nabla F = \mathbf{e}_x \frac{\partial F}{\partial x} + \mathbf{e}_y \frac{\partial F}{\partial y} + \mathbf{e}_z \frac{\partial F}{\partial z}$$

and

$$\nabla \cdot \mathbf{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z},$$

in polar coordinates

$$\nabla F = \mathbf{e}_r \frac{\partial F}{\partial r} + \frac{\mathbf{e}_\theta}{r} \frac{\partial F}{\partial \theta} + \frac{\mathbf{e}_\psi}{r \sin\theta} \frac{\partial F}{\partial \psi}$$

and

$$\nabla \cdot \mathbf{F} = \frac{1}{r^2} \frac{\partial \left(r^2 F_r\right)}{\partial r} + \frac{1}{r \sin\theta} \frac{\partial \left(\sin\theta F_\theta\right)}{\partial \theta} + \frac{1}{r \sin\theta} \frac{\partial F_\psi}{\partial \psi},$$

and in cylindrical coordinates

$$\nabla F = \mathbf{e}_{r_{cyl}} \frac{\partial F}{\partial r_{cyl}} + \mathbf{e}_z \frac{\partial F}{\partial z} + \frac{\mathbf{e}_\psi}{r_{cyl}} \frac{\partial F}{\partial \psi}$$

and

$$\nabla \cdot \mathbf{F} = \frac{1}{r_{cyl}} \frac{\partial \left(r_{cyl} F_{r_{cyl}}\right)}{\partial r_{cyl}} + \frac{\partial F_z}{\partial z} + \frac{1}{r_{cyl}} \frac{\partial F_\psi}{\partial \psi}.$$

Domain

Let Ω denote a finite spherical domain of radius R_D. The position vector of the centre of Ω is $\mathbf{r}_{\Omega_0}$, and generally $\mathbf{r}_{\Omega_0} = (0,\, 0,\, 0)$. Ω is bounded by Γ.

Units

Throughout this thesis, unless specifically specified otherwise, SI and SI-derived units are used. Guidelines for the correct use of the International System of Units (SI) can be found at the Bureau International des Poids et Mesures, `http://www.bipm.org/en/si/`.

SI base units

Quantity	**Name**	**Symbol**
length	metre	m
mass	kilogramme	kg
time	second	s
temperature	Kelvin	K

SI derived units

Quantity	**Unit**		
	Name	**Symbol**	**Equivalent**
angle	radian	rad	--
force	Newton	N	kg ms^{-2}
pressure	Pascal	Pa	$kgm^{-1}s^{-2}$
energy	Joule	J	kg m^2s^{-2}
frequency	Hertz	Hz	s^{-1}

For clarity I note here that units of milliseconds are denoted ‘‘ms’’.

Symbols

The major symbols used in this thesis are listed here. For complete definitions the reader should refer to the main text. The symbols α, a, f, η, m, n, ξ, τ and ϕ, are used for a variety of purposes throughout this thesis, and are not listed below. A double dash (--) in the units column indicates a vector quantity, for which the components have various dimensions. Where the quantity is dimensionless the units column is left blank. I include only the most common subscripts.

Symbol	Description	Units
A	Area (of a computational cell face)	m^2
β_S	Isentropic compressibility	Pa^{-1}
γ	Ratio of specific heat capacities	
Γ, $\hat{\Gamma}$	Boundary of computational (real) and ghost domains	
c	Speed of sound	ms^{-1}
$\mathbf{C}$	Vector of source terms and transverse derivatives	--
c_v, c_p	Specific heat capacity at constant volume and pressure	$\mathrm{Jkg}^{-1}\mathrm{K}^1$
δr, $\delta\theta$	Radial and polar dimensions of computational cell	m, rad
δt	Time step	s
δV	Volume of a computational cell	m^3
δx	Length of computational cell in x-direction	m
d	Depth of air gun below sea surface	m
e	Specific internal energy	Jkg^{-1}
E	Total energy	J
$\mathbf{F}$, $\mathbf{G}$	Vector of conservative fluxes	--
$\hat{\mathbf{F}}$, $\hat{\mathbf{G}}$	Numerical flux approximations of $\mathbf{F}$ and $\mathbf{G}$	--
$\mathbf{F}_\mu$	Viscous terms in Navier-Stokes equations	--
g	Acceleration due to gravity	ms^{-2}
η_K	Kolmogorov length scale	m
h	Specific enthalpy	$\mathrm{m}^2\mathrm{s}^{-2}$
H	Specific enthalpy at bubble wall	$\mathrm{m}^2\mathrm{s}^{-2}$
i, j	Spatial indices in r- or x- and θ- directions respectively	
$\mathbf{I}$	Identity matrix	
λ	Characteristic wave speed	ms^{-1}
$\mathscr{L}$	Characteristic wave mode	--
L	Spatial operator	
L_{turb}	Turbulent length scale	m
μ	Dynamic viscosity	$\mathrm{kgm}^{-1}\mathrm{s}^{-1}$
m	Number of computational cells in r- or x- direction	
Ma	Mach number	
p	Pressure	Pa
p_∞, ρ_∞	Pressure and density in undisturbed water	Pa, kgm^{-3}
P or P_{int}	Effective bubble interface pressure	Pa
p_c	Tamman EoS stiffener	Pa
q	Index of cell immediately left of interface in 1D model	
ρ	Density	kgm^{-3}
R or R_{int}	Effective bubble radius	m
R_D	Radius of Ω	m
R_{ss}	Sea-surface reflection coefficient	
Re	Reynolds number	
S	Characteristic wave speed	m^{-1}
$\mathbf{S}$, $\mathbf{D}$	Geometric and gravitational source terms	--

Symbol	Description	Units
τ_μ	Viscous stress tensor	Pa
$t_{collapse}$	Period of first bubble oscillation	s
t_G	Time at which signals reaching Ω at t left $\hat{\Omega}$	s
T	Temperature	K
T_{turb}	Turbulent time scale	s
$\mathbf{u}$	Velocity	ms^{-1}
u, v	Radial and polar velocity components, u_r and u_θ	ms^{-1}
$\mathbf{u}_{LS}$	Level-set advection velocity	ms^{-1}
$\mathbf{U}$	Vector of conservative properties	--
$\tilde{\mathbf{U}}$	Vector of primitive properties	--
V	Volume	m^3
V_b	Volume of bubble	m^3
V_i or $V_{i,j}$	Volume of computational cell	m^3
Ω, $\hat{\Omega}$	Computational (real) and ghost domains	
z_b	z-coordinate of centre of mass of bubble	m

Acronyms

Acronym	Description
ALE	Arbitrary Lagrangian-Eulerian
AMR	Adaptive Mesh Refinement
AUSM	Advective Upstream Splitting Method
CFL	Courant-Friedrichs-Lewy number
ENO	Essentially Non-Oscillatory
EoS	Equation of State
GFM	Ghost Fluid Method
HLL	Harten-Lax-van Leer
HLLC	Harten-Lax-van Leer with Contact restoration
LF	Lax-Friedrichs
LLF	Local Lax-Friedrichs
mGFM	modified Ghost Fluid Method
NLAA	Non-Linear Acoustic Approximation
NR	Non-Reflecting
PDE	Partial Differential Equation
PML	Perfectly Matched Layer
rGFM	real Ghost Fluid Method
SPH	Smoothed Particle Hydrodynamics
TV	Total Variation
TVD	Total Variation Diminishing
WENO	Weighted Essentially Non-Oscillatory

Chapter 1

Introduction

1.1 Background

Marine seismic exploration is a method employed by the hydrocarbon industry to find geological structures in the sub-surface with the potential to contain trapped hydrocarbons. The process is effectively a very powerful form of echo sounding. A source of seismic energy is towed behind a ship. The energy produced by the source propagates as a sound wave through the sea into the sub-surface. Within the sub-surface the energy is reflected, refracted and diffracted. The ship also tows an array of hydrophones behind the seismic source, and these are used to measure the wavefield. The measurements are digitised, transmitted to the vessel, and recorded. A schematic diagram of this configuaration is shown in Figure 1.1. If the source signal is known, then the received signal at each hydrophone can be deconvolved for the source signal to obtain the impulse response of the earth between the source and the hydrophone. These impulse responses can highlight some of the structures in the subsurface. Maps of the subsurface built up from these impulse responses are then interpreted to estimate the locations of trapped hydrocarbons.

In early marine seismic exploration dynamite was used as the source of seismic energy.

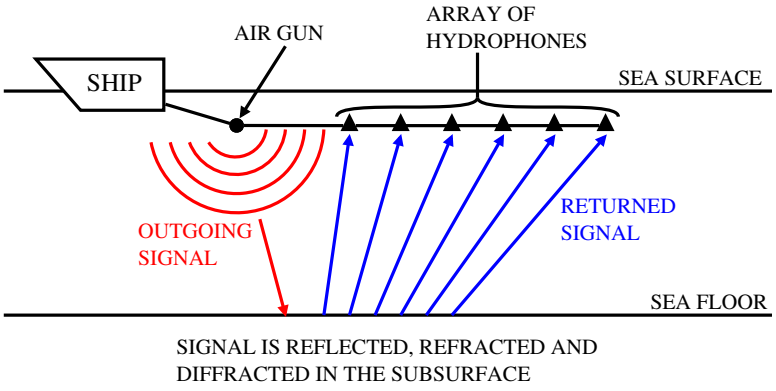


Figure 1.1: A schematic diagram of marine seismic exploration.

However, environmental concerns led to the development of a variety of alternative mechanical sources. By far the most common source in commercial use today is the marine seismic air gun. The marine seismic air gun, henceforth referred to simply as an air gun, is a steel cylinder which contains highly compressed air (typically up to 2000psi or 1.38×10^7Pa), towed between 5 and 20 metres below the sea surface. Air guns typically have volumes between about 20 and 250 cubic inches. When an air gun is fired, its ports open and the air is released into the water, forming a bubble, and transmitting a pressure pulse into the water. Due to the momentum of the water, the bubble expands beyond equilibrium, until the pressure in the bubble is lower than that in the water. The bubble then collapses, again beyond equilibrium, before expanding again. This process repeats, and the bubble oscillates until all the energy of motion is dissipated. Figure 1.2 shows pictures of an air gun bubble at various stages of oscillation. As the bubble oscillates, it also rises through the water due to buoyancy. Each time the bubble collapses a pressure pulse is produced which propagates out into the water. It is this series of pulses which constitutes the signal that propagates into the sub-surface. Air-gun bubble oscillations typically have a period of about 0.1 seconds, depending on the size, depth and pressure of the air gun. Figure 1.3 shows a pressure measurement taken 1.5 metres from an air gun. The wavelengths present in the signal produced by an air gun are much greater than the maximum diameter of the bubble. Hence, at distances greater than about 1 metre from the air gun, the bubble may be considered an acoustic monopole (Ziolkowski and Johnston, 1997). The signal produced by an air

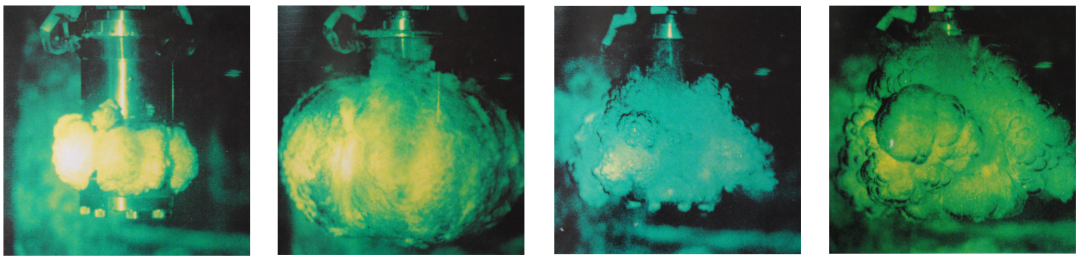


Figure 1.2: Photographs of an air-gun bubble at various stages of oscillation. (A) - shortly after firing; (B) - near maximum expansion; (C) - near the end of collapse; (D) - during second expansion. Adapted from Langhammer (1994).

gun reflects off the sea surface and interacts with the air-gun bubble. This reflected wave is known as the ghost. The effect of the ghost is visible in Figure 1.3 in the form of the sharp minimum at 0.02 seconds. Air guns are commonly deployed in arrays, with many guns of different sizes being fired at different times to tune the overall signal. In this case, the bubbles produced by the air guns also interact with each other. When air guns are deployed in arrays the arrays are often large enough that the signal becomes directional. A good history of marine seismic exploration and air guns is given in the introduction of Langhammer (1994).

There are two methods used by industry to determine the signal produced by an air gun or air gun array: (1) modelling, and (2) extrapolation from near-field measurements. Traditionally, industry uses the first method. With broader bandwidth data that are being recovered by removing the sea-surface reflection at the source and receiver (source and receiver ghosts) it has been found (Poole and Davison, 2013) that modelling is inferior to extrapolation from near field measurements, although industry has been slow to adopt the second method. Despite this change, modelling remains a valuable tool in the design of air gun arrays, where designs can be optimised by adjusting parameters of the array and using modelling to determine the wavefield of each variation of the array. The aim of this work is to develop methods which can improve on current air-gun bubble modelling.

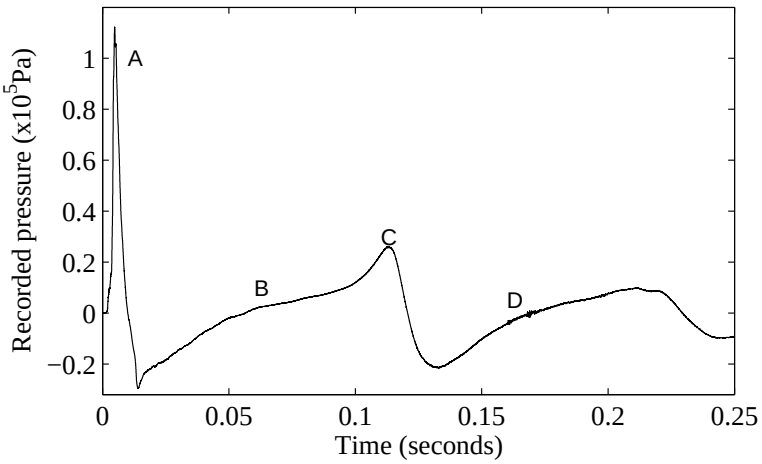


Figure 1.3: Pressure measurements taken 1.5 metres from an air gun. Note that the measurements are relative to hydrostatic pressure (approximately 1.7Bar), and that there a no negative absolute pressures. The points marked 'A' to 'D' correspond approximately to the images in Figure 1.2. The data were provided by Petroleum Geo-Services.

1.2 Review

Research on the behaviour of oscillating bubbles has a long history. The motion of a spherical cavity in a liquid was first investigated by Rayleigh (1917), who developed an equation for the motion assuming the water to be incompressible. In this work, the cavity or bubble is described by a time-varying radius, $R(t)$ and pressure, $P(t)$. Lamb (1923) developed an exact wave equation to describe the motion of a spherical bubble. During the Second World War, underwater explosions were an important area of research, as an understanding of the damage caused to ships and submarines by underwater explosions was crucial in the design of both better ships and submarines, and better explosives. Several authors extended the work of Rayleigh (1917) and Lamb (1923) during the war, for instance Herring (1941) and Kirkwood and Bethe (1942). Herring (1941), Taylor and Davies (1943) and Taylor (1942) also conducted research during the Second World War into the shape and motion of oscillating bubbles. After the Second World War, much of the work on underwater explosions by British and American researchers was published in the three volumes of "Underwater Explosion Research" (1951), the second volume of which is dedicated specifically to the study

of the bubble produced, referred to as ‘‘the gas globe’’. Another useful work on the theory of underwater explosions prior to 1948 is ‘‘Underwater Explosions’’ by Cole (1948). Gilmore (1952) extended the work of Kirkwood and Bethe (1942) to obtain an improved approximate equation of motion for the bubble. Keller and Kolodner (1956) investigated the mechanisms by which underwater explosion bubble oscillations are damped.

Another important application of oscillating bubbles is cavitation. This is the phenomenon which occurs when the pressure in a liquid drops to such a level that cavities form, referred to as cavitation bubbles. These cavities are unstable and collapse, generating very high pressures and temperatures. The repeated high pressures associated with cavitation bubble collapse can cause serious wear to metal. The most prevalent example of cavitation damage is the damage caused to pump impellers and ships propellers. Cavitation can also have a negative effect on the efficiency of pumps. Work on cavitation bubbles continued from the pre-1950 work on underwater explosions (see Plesset (1949), Plesset and Zwick (1952), Prosperetti and Plesset (1978), Blake and Gibson (1981), Tomita and Shima (1986), Prosperetti and Lezzi (1986) and Lezzi and Prosperetti (1987) for example), leading to the widely used Rayleigh-Plesset class of approximations for oscillating bubbles.

More recently, many authors have developed finite volume methods (an introduction to which is given in the next chapter) (for example Flores and Holt (1981), Wardlaw and Mair (1998), Smith (1999), Hu *et al.* (2006), Pishevar and Amirifar (2010), Shaw and Spelt (2010), Barras *et al.* (2012) and Miller *et al.* (2013)) and other methods, such as boundary element methods and boundary integral methods (for example, Kucera and Blake (1990), Wilkerson (1992) and Blake *et al.* (1999)) for the simulation of underwater explosions and cavitation bubbles. In these works, emphasis has predominantly been on the treatment of the gas-water interface. There is now a significant field of research and commercial software dedicated to underwater explosions (often referred to as UNDEX).

Whilst the field of numerical simulation of oscillating bubbles has developed fairly continuously over the past 60 years, authors modelling air-gun bubbles seem to have

been, for the most part, either unaware or uninterested in these developments. Most air gun modelling follows the research on oscillating bubbles up to Gilmore (1952), before separating, and thereafter referencing only other authors within the hydrocarbon industry.

Air guns were first modelled by Ziolkowski (1970), based on the early work of Rayleigh (1917), Lamb (1923), Kirkwood and Bethe (1942) and Gilmore (1952). Ziolkowski (1970) presented a model which consists of an ordinary differential equation in $R(t)$ and $P(t)$, and a polytropic equation of state for the contents of the bubble. A schematic diagram of this model is shown in Figure 1.4. The bubble is assumed to be spherical and homogeneous. Various approximations allow the pressure away from the bubble to be calculated. Dragoset (1984) described an alternative model for air-gun bubbles, including viscous damping terms to obtain better results, although complete details of the model are omitted. Laws *et al.* (1990), Landrø (1992) and Landrø and Sollie (1992) used the model of Ziolkowski (1970) and included viscous terms to provide more damped results, some of which were based on an earlier method developed by Bornhorst and Hatsopoulos (1967) for caviation bubbles. Johnson (1994) developed a model for air-gun bubble oscillations based on an analogy with a damped mass-spring system. Ziolkowski (1998) presented a method to calculate the wavefield produced by an air gun from near-field measurements. This method contains an improvement to the equation of motion used in the model of Ziolkowski (1970), which removes the need for a complex transition between the near-field and the far-field. It also allows the pressure measurement to be made much closer to the bubble centre - even inside the bubble. The advantage of this is that the pressures are much higher and the influence of waves from other guns is much less. In principle it is a more precise method than that of the earlier paper. To date, the improvment of Ziolkowski (1998) does not appear to have been taken up by industry. It should also be noted that the approximation of Ziolkowski (1998) was first described, although in a different form, by Herring (1941). Li *et al.* (2010) carried out further investigations on the model of Ziolkowski (1998), including a more complex equation of state for the contents of the bubble.

Since Ziolkowski (1970) it has been known that air-gun bubble models show less

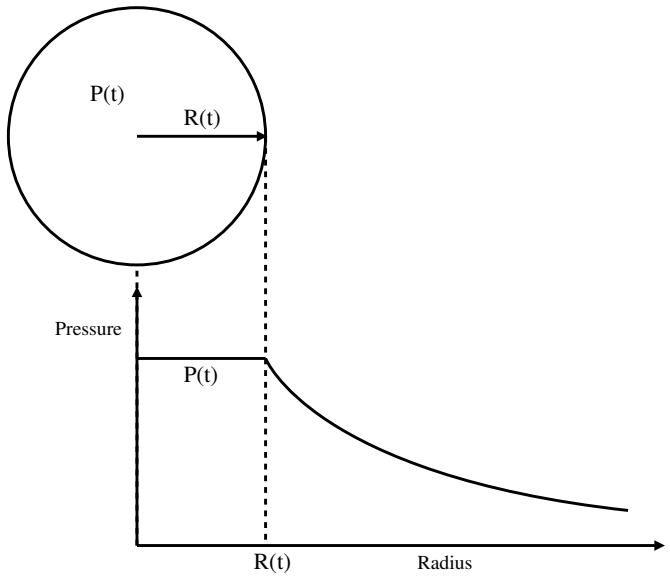


Figure 1.4: A schematic diagram of a traditional air-gun bubble model. The bubble is assumed to be spherical, with uniform internal pressure. Various approximations allow the pressure field outside the bubble to be calculated. In its most simple form, the entire system is described by the two time-varying parameters $R(t)$ and $P(t)$.

damping than measurements. There are various processes which cause this damping. The primary damping mechanism is the compressibility of the water, which allows for the conversion of the energy in the bubble into heat in the water (Note that due to the high thermal conductivity of water and its large bulk, the temperature in the water is approximately constant.). The surfaces of air-gun bubbles are known to be non-smooth, and this presents another damping mechanism. The surfaces themselves have an associated surface energy, and the increased surface area increases the rate at which heat and mass transfer can occur. As air-gun bubbles oscillate they rise, and in doing so lose energy to the water through hydrodynamic drag. Air-gun bubbles also deform, and it is known (Cox *et al.*, 2004) that during collapse jets form on the underside of bubbles. These jets present yet another energy sink which damps bubble oscillations. Viscosity also plays a role, both dissipating energy directly, and increasing the drag on rising bubbles. Figure 1.5 depicts the jet as a bubble collapses and the effect of viscosity in the velocity field around a bubble.

Many of the models referenced above attempted to obtain better results by developing

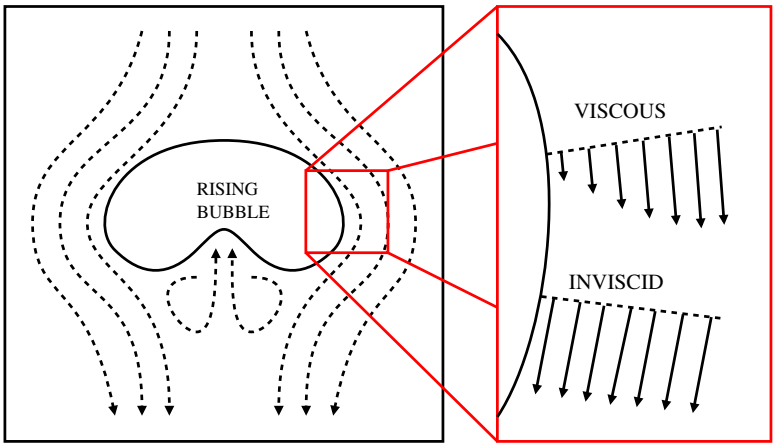


Figure 1.5: An illustration of the streamlines associated with jet formation as a rising bubble collapses. The magnified section on the right shows the effect of viscosity on boundary layers around the bubble, which increase the drag as the bubble rises. The length of the arrows represents the magnitude of the velocity of the water relative to the bubble.

rather vaguely justified systems of equations to describe the equation of state of the bubble contents. In order to provide results which agreed well with data, various damping mechanisms were incorporated, such as the transfer of energy between the bubble and the water by way of evaporation at the bubble wall and the release of latent heat when vapour condensed in the bubble. Ziolkowski and Metselaar (1984) outlined one such model for the equation of state which contain a 'surface area factor' (see Appendix D). This factor increases the rate of diffusion at the bubble wall by a factor of about 600 in order to provide the necessary damping to bubble oscillations, whilst assuming the bubble remains spherical. The phenomena included in this model are shown in Figure 1.6. It is clear from photographs of air-gun bubbles (Langhammer (1994), Langhammer and Landrø (1996) and Figure 1.2) that the bubble surfaces are not smooth. They in fact appear more like a foam, composed of many smaller bubbles of varying sizes, and hence must have a surface area greater than that of a smooth sphere of the same volume. However, there is no way of quantifying how the surface area compares with that of a smooth spherical bubble, and it certainly varies throughout the bubble oscillation. The model of Ziolkowski and Metselaar (1984) was never fully published, though I have reconstructed it based on private communication with the first author and an unpublished report (Ziolkowski, 1986), and it is described fully

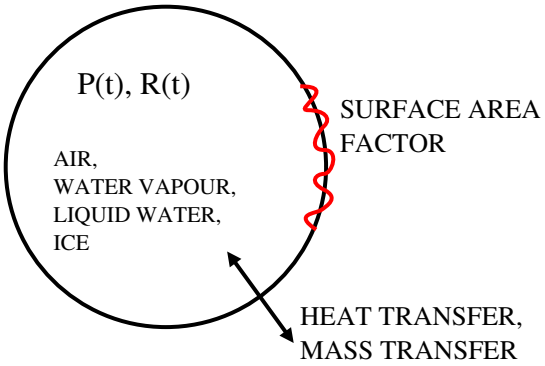


Figure 1.6: An illustration of the effects included in the model of Ziolkowski and Metselaar (1984). The bubble is assumed to be spherical and homogeneous, containing air, water vapour, liquid water and ice. The non-smooth surface is accounted for with a surface area factor.

in Appendix D. All air gun modelling referenced above is based on the two-variable framework of Ziolkowski (1970) or earlier work of Rayleigh (1917), Herring (1941) and Gilmore (1952). These models are based on the assumption that the bubble is spherical, and the contents of the bubble are homogeneous. Attempts to include more complex effects, such as the presence of ice in the bubble, or the wrinkly nature of the bubble surface violate the fundamental assumptions on which the models are based.

Langhammer (1994) provided a clear description of the models based on Ziolkowski (1970), and is a useful reference for the history of air guns and air-gun bubble modelling. Langhammer and Landrø (1993a) investigated the effects of viscosity on air-gun bubbles, primarily focussing on the effects of viscosity on the wavefield emitted by the bubble. Langhammer and Landrø (1993a) showed that the effect of viscosity on air gun signatures was in the opposite direction to the effects of the viscous terms introduced by Bornhorst and Hatsopoulos (1967), and used in simulations of air guns by Landrø (1992) and Dragoset (1984). Viscosity would have to be negative for these terms to give the effect attributed to them. Langhammer and Landrø (1993a) mention turbulent motion briefly, as a damping mechanism, but did not elaborate. Langhammer and Landrø (1993b) investigated the effects of water temperature on air gun signatures. Langhammer *et al.* (1995) performed a holographic study of air-

gun bubbles to investigate the smoothness of the air-water surface. Langhammer and Landrø (1996) presented photographs of air-gun bubbles using high speed photography. Whilst these photos remain a valuable resource, they did not observe the formation of jets, and agree with the work of Herring (1941), Taylor and Davies (1943) and Taylor (1942) that the bubble remains nearly spherical for several oscillations.

The calculation of the wavefield produced by an array of air guns was first considered by Giles and Johnston (1973). Further considerations on array design were given by Brandsaeter *et al.* (1979). A method to calculate the far-field signatures of arrays from near-field measurements was developed by Ziolkowski *et al.* (1982) and Parkes *et al.* (1984). The near-field is in the linear zone, where the $1/r^4$ term in the radiated pressure field is negligible. This is at distances greater than about 1 metre from the bubble centres. Recently, methods have been developed in Barker and Landrø (2012, 2013, 2014) to model clusters of air guns very close together, using the concept of isosurfaces of velocity potential to model several coalescing bubbles as a single bubble.

Cox *et al.* (2004) reviewed several different air-gun bubble models, including one using boundary element methods similar to Blake *et al.* (1999). Their models show bubble deformation, although they do not capture surface instabilities. Cox *et al.* (2004) also investigated the rate at which air-gun bubbles rise, and compared the results of simulations with experimental data. To the best of my knowledge, there have been no published finite volume simulations of air-gun bubbles, or models in which the contents of the bubble are inhomogeneous.

For problems of oscillating bubbles there can be two orders of magnitude difference or more between the speed of pressure waves and the speed of the air water interface. During one oscillation of an air-gun bubble, the pressure wave might propagate 150 metres outwards (300 times the maximum bubble radius). Hence, for finite volume simulations of bubbles the computational domain required to capture the entire problem is very large. One approach to the problem is to truncate the domain and apply an artificial boundary condition on the boundary. The purpose of an artificial boundary condition is to mimic the behaviour of everything outside the computational domain.

The application of an appropriate artificial boundary condition is not trivial. For spherical oscillating bubbles, the solution contains a weak incoming wave, which must be accurately described by the boundary conditions in order to avoid the contamination of results.

Boundary conditions are an important problem in many branches of physics, and have been studied extensively since the advent of computers brought numerical simulations to the fore. Good reviews of previous research into artificial boundary conditions are given by Givoli (1991), Tsynkov (1998), Hagstrom (1999) and Colonius (2004).

Early work on artificial boundary conditions focussed on the wave equation, for example Engquist and Majda (1977), Bayliss and Turkel (1980), Grote and Keller (1995). A review of early work on linear hyperbolic systems is given by Higdon (1986). Hedstrom (1979) decomposed the Euler equations into characteristic wave equations, and presented a non-reflecting boundary condition using these equations. Thompson (1987) and Thompson (1990) presented a useful formalism for the application of artificial boundary conditions, based on the decomposition of Hedstrom (1979). Another approach initially developed around the wave equation is the perfectly matched layer (PML) method (Berenger, 1994, 1996; Abarbanel and Gottlieb, 1997; Turkel and Yefet, 1998), in which a buffer region is placed around the computational domain, in which outgoing waves at attenuated. Similar methods have been developed for the Euler equations, for example Hayder *et al.* (1999). Artificial boundary conditions have been designed by linearisation of the Euler equations (Hagstrom and Goodrich, 2003), and also by considering the asymptotic solution in the far-field (Hagstrom and Hariharan, 1988). Other authors (Atassi and Galan, 2008) have developed artificial boundary conditions for inflow and outflow conditions with vortical disturbances. With the exception of Hagstrom and Hariharan (1988), which as presented is limited to isentropic motion, none of these methods are capable of capturing the weak incoming wave present in spherically symmetric problems.

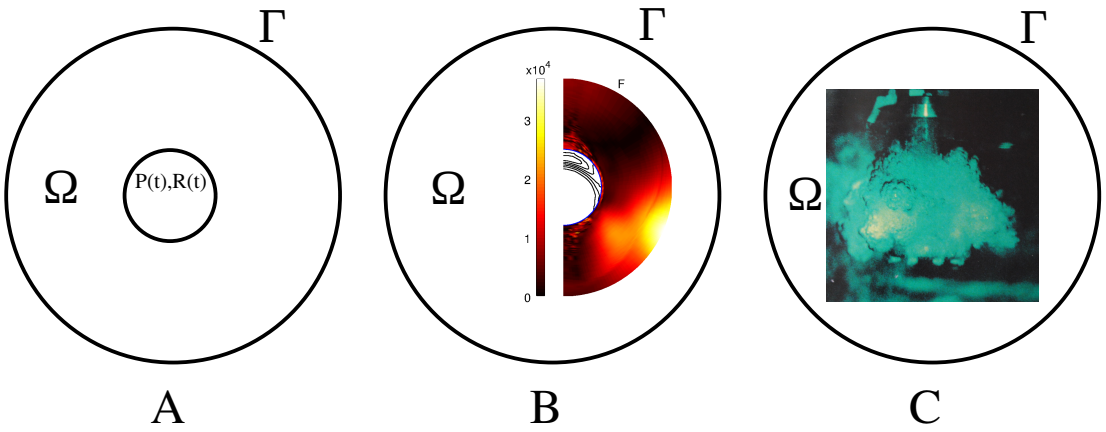


Figure 1.7: An illustration of my approach to air-gun bubble modelling. The domain is denoted Ω and the domain boundary Γ. In Ω any set of assumptions may be adhered to by the model. (A) - one-dimensional spherically symmetric model; (B) - two-dimensional axisymmetric model; (C) - the limiting case of a perfectly realistic model (image adapted from Langhammer (1994)). This illustration demonstrates the type of information each model would provide.

1.3 Claim

The purpose of this thesis is to improve on current air-gun bubble modelling, and to further our understanding of the physical processes which occur in air-gun bubbles. From a comparison of Figures 1.2 and 1.4 it is obvious that traditional air-gun bubble models represent a greatly simplified version of reality. Physical processes such as compressibility, viscosity, turbulence, surface tension, surface instabilities, bubble deformation, bubble translation and heat and mass transfer all play a role, to differing degrees, in the motion of the bubble. Traditional air-gun bubble models are based on the assumption that the motion of the water is of low Mach number, the bubble is spherically symmetric, and that viscosity is negligible. The presence of many of the physical processes mentioned above violates the assumptions on which the traditional air-gun bubble models are based.

My approach is different to traditional models. I consider a computational domain, denoted Ω, larger than and completely enclosing an air-gun bubble. Within the domain

the air-gun bubble is modelled, with any desired set of assumptions regarding the physics. This is illustrated in Figure 1.7. Frame (A) shows a one-dimensional model within the domain. Frame (B) illustrates a two-dimensional model. Frame (C) depicts a ‘‘perfectly realistic’’ model within the domain. Whilst the model in frame (C) is clearly unfeasible due to constraints on computing power, there is no other reason to prevent a model of this form from being developed. This is in contrast with traditional models. Outside the domain, the same assumptions as in traditional models are applied. Whilst the assumptions on which traditional models are based are not strictly valid on the bubble surface, they are valid away from the bubble. The domain is large enough that these assumptions are valid on the domain boundary, denoted Γ. At the domain boundary, there must be some mechanism to allow the two regions to interact. Incorrect artificial boundary conditions may cause spurious reflections at the domain boundary, which can propagate through the domain contaminating results. This is illustrated in the left panel of Figure 1.8. My primary contribution is a novel artificial boundary condition which facilitates a model of this type. The boundary condition takes information about the motion from within the computational domain, and calculates an approximation of the motion outside the domain. This information is then passed back into the domain as a boundary condition. This is illustrated in the right panel of Figure 1.8. Through this approach finite volume simulations of air gun bubbles, which present much greater scope for flexibility of assumptions than traditional models, may be conducted on a small computational domain, hence reducing computational costs.

I develop a finite volume scheme which I use in conjunction with the artificial boundary condition. In my models the contents of the bubble are not considered uniform. In the two-dimensional model the bubble is not constrained to be spherically symmetric, and the model captures the effects of viscosity, surface instabilities, bubble translation and bubble deformation. The boundary condition performs well, allowing finite volume simulations of bubbles to be carried out for long run-times on comparatively small domains, reducing computational costs. The one- and two-dimensional simulations of air-gun bubbles I present are, to my knowledge, the first finite volume simulations of air-gun bubbles carried out. They elucidate non-spherical aspects of air-gun bubble

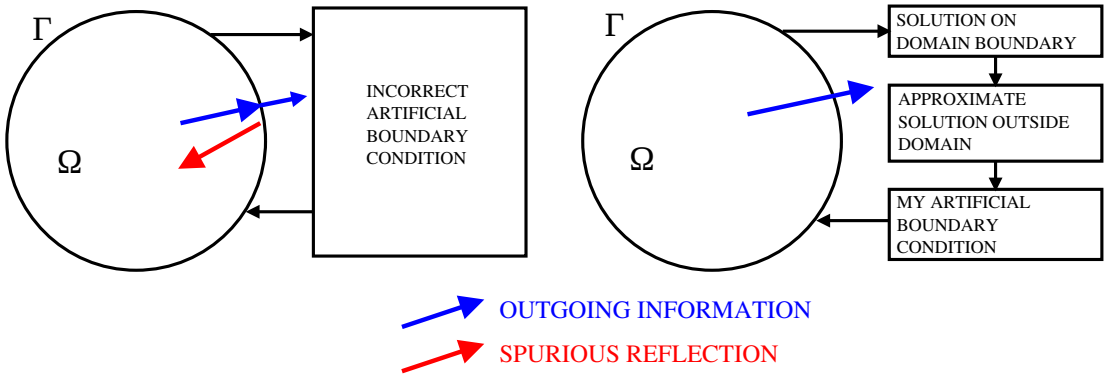


Figure 1.8: A conceptual illustration of artificial boundary conditions. Left panel: incorrect boundary conditions causing reflections at the domain boundary. Right panel: my artificial boundary condition.

behaviour, which may in due course be incorporated into the models used by industry. The results of the two-dimensional air gun modelling highlight the potential value of my artificial boundary condition, and also highlight the aspects of my computational scheme which require improvment. I extend the scheme to include the effects of the sea surface, increasing the practical value of my contribution, and leading to a novel suggestion of a mechanism for the production of higher frequencies which are observed in data.

1.4 Agenda

The layout of this thesis is as follows. In Chapter 2 I give some background and theory to the numerical methods for solving hyperbolic partial differential equations used in this thesis. In Chapter 3 I derive a novel artificial boundary condition for finite volume simulations of oscillating bubbles. In Chapter 4 I present the details of the numerical scheme I have developed which I use for all finite volume simulations in this thesis. Chapter 5 contains one-dimensional results of finite volume modelling of air-gun bubbles and underwater explosions, using my novel boundary condition. I demonstrate some of the benefits and limitations of the numerical scheme, and quantify the errors introduced by the boundary condition. Chapter 6 contains the results of two-

dimensional finite volume modelling of air-gun bubbles. I demonstrate that the artificial boundary condition developed in Chapter 3 is effective even in situations without spherical symmetry. I discuss the errors introduced by the boundary condition, and the limitations of the numerical scheme in general. Chapter 6 also contains a discussion of the various bubble phenomena present in the results, such as surface instability and jet formation, and their impact on far-field bubble signatures. In Chapter 7 I describe a method to include viscous terms in the model, and hence solve the full compressible Navier-Stokes equations. I present results of viscous and inviscid models, and discuss the effects of viscosity on air gun signatures. In Chapter 8 I show how the scheme of Chapters 3 and 4 may be extended to include the effect of the sea surface. I give details of the numerical method, and provide results showing the impact of the sea surface on the bubble. I discuss the possibility of a more general extension to simulate arrays of air guns and more complex geometries. I present results which give new insight into previously unexplained aspects of measurements. In Chapter 9 I draw conclusions on the work in the thesis, and discuss the benefits and limitations of my approach. I highlight the aspects of my approach which require improvement.

Appendix A contains details of additional numerical procedures used in the scheme described in Chapter 4. Appendix B describes the poorly documented problem of pressure discontinuities at the interface often encountered in numerical simulations of oscillating bubbles based in non-Cartesian coordinate systems. Appendix C contains a derivation of the criteria for the stability of an air-water interface. Appendix D contains the details of a scheme for air-gun bubble modelling typical of the schemes used by industry. Appendix E contains a listing of the source code developed during this research.

1.5 Publications arising from the thesis

The work in Chapters 3, 4, 5 and parts of Chapter 6 has been published in the *Journal of Computational Physics*, under the title *Boundary conditions*

for simulations of oscillating bubbles using the non-linear acoustic approximation (King *et al.*, 2015), and can be found online at the following URL: `http://dx.doi.org/10.1016/j.jcp.2014.12.037`.

Chapter 7 has been submitted to *Geophysics* under the title *Viscosity in air-gun bubble modelling* and is currently undergoing peer review. Chapter 8 has been submitted to *Geophysics* under the title *Air-gun bubble-ghost interactions*, and is currently undergoing peer review.

The published paper and both unpublished manuscripts are appended to this thesis.

Chapter 2

Numerical methods for hyperbolic PDEs

2.1 Introduction

In this thesis I develop a new finite volume scheme for air gun bubble oscillations. The scheme solves the Euler equations, which are a system of non-linear hyperbolic partial differential equations (PDEs). In this chapter I provide an introduction and some background to finite difference and finite volume schemes for hyperbolic PDEs. Throughout this thesis I consider Eulerian schemes, in which the equations are solved on regular fixed grids. For clarity, in this chapter I describe the methods in one spatial dimension x only.

The layout of this chapter is as follows. In Section 2.2 I introduce a simple hyperbolic PDE, discuss the form of its solution and describe how it is solved using a finite difference scheme. In Section 2.3 I introduce the concepts of monotonicity preservation and total variation, and describe the importance of these concepts for the stability of a numerical scheme. In Section 2.4 I outline how the simple hyperbolic PDE may be solved using a finite volume scheme. In Section 2.5 I describe reconstruction procedures

used in finite volume schemes. In Section 2.6 I introduce the concept of a Riemann problem, and discuss the applications of Riemann problems and methods of solution. In Section 2.7 I discuss the various numerical fluxes developed for use in finite volume schemes. In Section 2.8 I briefly review some high-order time integration methods.

2.2 Finite difference schemes for hyperbolic PDEs

Consider the first-order hyperbolic partial differential equation in ϕ

$$\frac{\partial \phi}{\partial t} + a\frac{\partial \phi}{\partial x} = 0, \tag{2.1}$$

where a is a constant. Let $\phi_0 = \phi\left(x, t = 0\right)$. By inspection, it can be seen that the solution to equation 2.1 is

$$\phi\left(x, t\right) = \phi_0\left(x - at\right). \tag{2.2}$$

The solution to equation 2.1 propagates along the x-axis with speed a without change in shape. Because of the simple solution to equation 2.1, it is a good prototype hyperbolic PDE with which to demonstrate the principles of the numerical methods underpinning this work.

I wish to solve equation 2.1 numerically, together with the initial data

$$\phi_0 = \begin{cases} 1 & -0.5 \leq x \leq 0, \\ 0 & 0 < x \leq 0.5, \end{cases} \tag{2.3}$$

and $a = 1$. This describes a step profile, with a discontinuity at $x = 0$, which is advected along the x-axis with a velocity a. The computational domain is defined as the set of points $x \in [-0.5, 0.5]$, for $t \geq 0$. The domain is represented by a set of m discrete points x_i, for $i \in [1, m]$, where $x_i = (i-1)\,\delta x - 0.5$ and $\delta x = 1/(m-1)$. A typical finite difference scheme to solve equation 2.1 is structured as follows. The solution at time t is known, and approximated by the set of discrete values ϕ_i^n, $i \in [1, m]$. For all i, (1) an approximation of the spatial derivative $\left.\frac{\partial \phi}{\partial x}\right|_i^n$ is calculated; (2) $\left.\frac{\partial \phi}{\partial x}\right|_i^n$ is substituted into equation 2.1 to estimate time derivative $\left.\frac{\partial \phi}{\partial t}\right|_i^n$ and (3) $\left.\frac{\partial \phi}{\partial t}\right|_i^n$ is integrated to obtain the solution ϕ_i^{n+1} at $t + \delta t$.

The focus of this section is on the methods by which the spatial derivatives are calculated. For simplicity, I use the first-order forward Euler method for time integration

$$\phi_i^{n+1} = \phi_i^n + \delta t \left.\frac{\partial \phi}{\partial t}\right|_i^n . \tag{2.4}$$

I discuss higher-order time integration methods in Section 2.8. The time step δt is constrained by the Courant condition, also known as the Courant-Friedrichs-Lewy condition (Courant *et al.* (1928) and Courant *et al.* (1967)), such that information cannot propagate more than a distance δx in the time δt. In this simple example case, the time step is set by the equation

$$\delta t = \mathrm{CFL}\delta x/a, \tag{2.5}$$

where CFL is known as the CFL number or sometimes Courant number. A stable simulation requires that $0 < \mathrm{CFL} \leq 1$. Whilst for equation 2.1 the maximum propagation speed of the solution is a, for many hyperbolic PDEs it is a function of the solution itself. In these cases, the value of the time step may be updated every time a new solution is obtained.

A first attempt at calculating $\left.\frac{\partial \phi}{\partial x}\right|_i^n$ might be the symmetric difference equation

$$\left.\frac{\partial \phi}{\partial x}\right|_i^n = \frac{\phi_{i+1}^n - \phi_{i-1}^n}{2\delta x}, \tag{2.6}$$

which can be shown with the use of Taylor expansions to be second-order accurate. However, this is not a good choice. The reasons for this can be seen by considering the nature of the solution to equation 2.1, and of the solution to hyperbolic PDEs in general.

Because the solution to equation 2.1 translates along the x-axis at speed a without change in form, it is a function only of $\xi = x - at$. In the x-t plane, lines along which ξ is constant are referred to as *characteristics*. As the solution is a function of ξ, which is constant along characteristics, so the solution solution is constant along characteristics. Whilst equation 2.1 has a single characteristic with a constant gradient in the x-t plane, more complex hyperbolic PDEs, such as the shallow water equations or the Euler equations have several characteristics, which are not necessarily straight, and

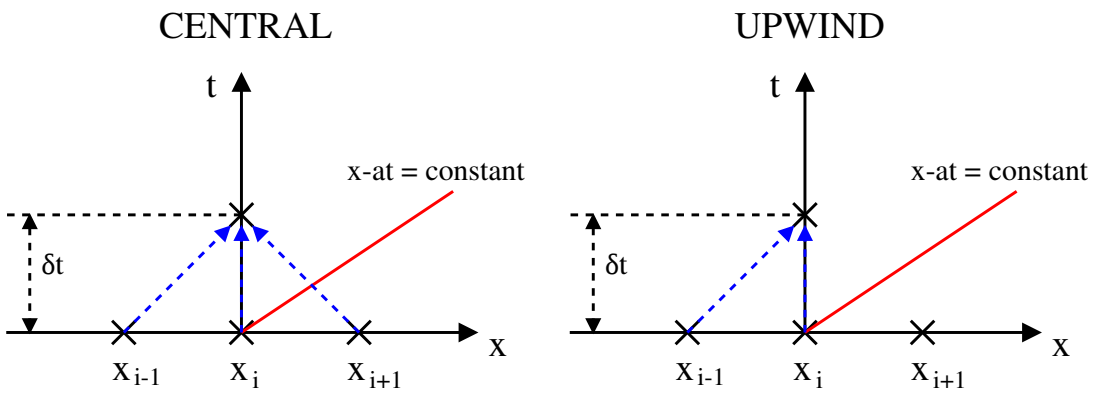


Figure 2.1: Second-order central differencing (left panel) and first-order upwind differencing (right panel) schemes for equation 2.1. The red line represents the characteristic. The dashed blue arrows represent the propagation of information.

have a gradient which is a function of the solution. For example, solutions to the Euler equations are described by two characteristics for the propagation of sound waves, and an entropy characteristic, which propagates with the local velocity. Whilst the entire solution of equation 2.1 is invariant along its characteristic, in more complex cases, different components of the solution are invariant along the different characteristics. The quantities which are invariant along a characteristic are called *Riemann invariants*. Returning to the example of the Euler equations, velocity and pressure are Riemann invariants of the entropy characteristic.

Characteristics describe the propagation of information through the solution space. Information about the solution propagates along the characteristics, and some information may not cross the characteristics. In the case equation 2.1, where a single characteristic describes the entire solution, no information may propagate across the characteristic. Equation 2.6 is not a good choice of difference scheme for equation 2.1 because it takes information from both sides of the characteristic. This is illustrated in the left panel of Figure 2.1. The solution obtained from a finite difference simulation using equation 2.6 is shown in red in Figure 2.2. The solution contains oscillations behind the discontinuity. These oscillations are unstable, and in due course the simulation will break down.

A better choice of discretisation for the spatial derivatives is the first-order *upwind* method given by

$$\left.\frac{\partial \phi}{\partial x}\right|_i^n = \frac{\phi_i^n - \phi_{i-1}^n}{\delta x}. \tag{2.7}$$

The propagation of information for this scheme is shown schematically in the right panel of Figure 2.1. The resulting solution with this discretisation method is shown in the blue trace in Figure 2.2. The term 'upwind' refers to the fact that the solution ϕ_i^{n+1} in cell i at $t + \delta t$ depends only the solution at time t in cell i and those cells upwind of it in the sense of the direction of propagation of the solution. There is no ringing behind the discontinuity in this solution, although the solution has been smoothed somewhat. The smoothing effect is unavoidable in finite difference schemes, due to truncation errors, although it can be reduced with different discretisation or time integration schemes. Using an upwind scheme to solve equation 2.3 is trivial, as the solution always propagates in the same direction with speed a. Other hyperbolic systems of PDEs, such as the Euler equations, have solutions in which the propagation speeds of waves are functions of the solutions themselves. In these cases, differencing schemes must contain switches which change the stencil depending on the local solution.

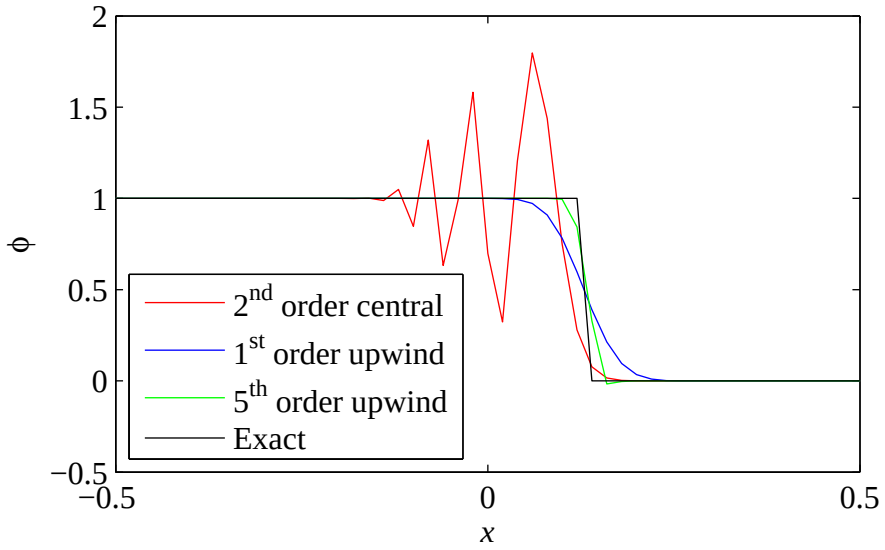


Figure 2.2: The solution to equation 2.1 at $t = 0.12$, using different differencing schemes. The black line shows the exact solution. All traces are calculated and plotted at a resolution of $\delta x = 0.02$.

When choosing a discretisation method for a finite difference scheme it is important to choose one in which the smoothing of shocks and discontinuities is minimised. Such schemes are often referred to as *shock capturing*, as they attempt to reduce dissipation such that shocks are accurately resolved. Through the manipulation of Taylor expansions and interpolation methods, many such high-order schemes are available. A class of such methods is essentially non-oscillatory (ENO) schemes. ENO schemes use a locally adaptive stencil to take information automatically from the smoothest regions when there are discontinuities present. The basis for ENO schemes was perhaps work by van Leer (1974, 1979). Much work was published on the implementation of ENO schemes by, for instance, Shu and Osher (1988), Harten (1989) and Shu and Osher (1989). Weighted ENO (WENO) schemes (Liu *et al.* (1994) Jiang and Shu (1996) and Balsara and Shu (2000)) are an extension of ENO schemes in which a weighted combination of stencils is dynamically chosen. This enables a higher order approximation to be used at smooth parts of the solution, and an upwind discretisation near shocks. For parts of this thesis I use a WENO scheme due to Borges *et al.* (2008), details of which I give in Appendix A. This scheme is fifth-order accurate in space. The solution to equation 2.1 at $t = 0.12$ using this scheme is shown in green in Figure 2.2. The speed of propagation matches the exact solution, and there is less smoothing of the discontinuity than for the first-order upwind scheme. ENO and WENO reconstruction procedures are not limited to finite difference schemes. They may also be used to reconstruct cell edge properties from which numerical fluxes may be calculated for use in finite volume schemes.

2.3 Monotonicity preservation and total variation diminishing schemes

The stability of a numerical scheme can be formally considered by introducing the concepts of monotonicity preservation and total variation. A numerical scheme to solve equation 2.1 is *monotonicity preserving* if, as time increases, no new local extrema of ϕ are created, the values of existing local minima do not decrease, and the values

of existing local maxima do not increase. An example of a scheme which is not monotonicity preserving is the unstable scheme described in equations 2.4 and 2.6, the results of which are shown in red in Figure 2.2.

The concept of total variation diminishing schemes was first introduced by Harten (1983). The total variation of a solution on a discrete set of points is given by

$$TV\left(\phi^n\right) = \sum_{i=1}^{m-1} \left|\phi_{i+1}^n - \phi_i^n\right|. \tag{2.8}$$

A scheme is said to be *total variation diminishing* (TVD) if

$$TV\left(\phi^{n+1}\right) \leq TV\left(\phi^n\right). \tag{2.9}$$

Harten (1983) proved that a total variation diminishing scheme is monotonicity preserving. A numerical scheme must be total variation diminishing in order to prevent the creation of spurious oscillations around discontinuities. This formalism is useful when designing some aspects of numerical schemes, such as time integration algorithms, where the TVD constraint may be enforced to obtain high-order time integration schemes which are subject to minimal CFL constraints.

2.4 Finite volume schemes

I now recast equation 2.1 in conservation form

$$\frac{\partial \phi}{\partial t} + \nabla \cdot \mathbf{F}\left(\phi\right) = 0, \tag{2.10}$$

where $\mathbf{F}\left(\phi\right) = a\phi\mathbf{e}_x$. Note that whilst I use the operator $\nabla\cdot$ in equation 2.10, I limit consideration to problems with variation in only one spatial dimension. Let the point x_i be the centre of the computational cell i, which extends from $x_{i-\frac{1}{2}}$ to $x_{i+\frac{1}{2}}$, and has volume δV_i. The points with index $i \pm \frac{1}{2}$ are the average position of each pair of adjacent points, such that $x_{i\pm\frac{1}{2}} = \left(x_{i\pm1} + x_i\right)/2$. Figure 2.3 shows a computational cell for a two-dimensional finite volume scheme. I show this in two dimensions because it makes visualisation of the concept simpler.

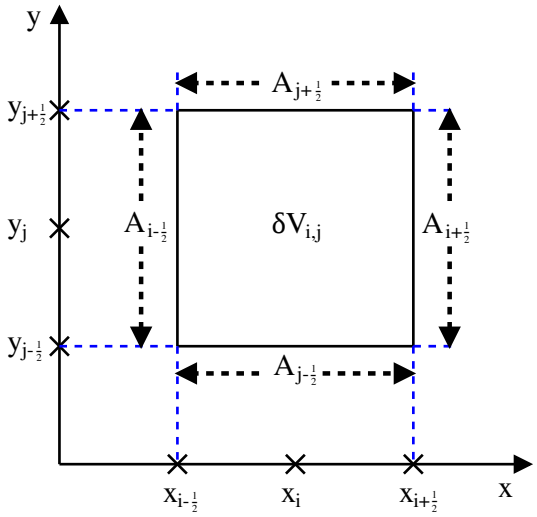


Figure 2.3: A computational cell for a two-dimensional finite volume scheme.

Equation 2.10 is integrated over the volume of each cell δV_i, to give

$$\int_{\delta V_i} \left(\frac{\partial \phi}{\partial t} + \nabla \cdot \mathbf{F}(\phi) \right) dV = \frac{\partial \phi}{\partial t} \delta V_i + \int_{\delta V_i} \nabla \cdot \mathbf{F}(\phi)\, dV = 0, \tag{2.11}$$

where, in Cartesian coordinates, the volume element $dV = dxdydz$. Note that in this one-dimensional example, $\delta V = \delta x$, but I refer to it as δV for clarity of origin. I denote the surface of the i-th computational cell S_i. Using Gauss' theorem equation 2.11 becomes

$$\frac{\partial \phi}{\partial t} \delta V_i + \int_{S_i} \mathbf{F}(\phi) \cdot \mathbf{n} dA = 0, \tag{2.12}$$

where $\mathbf{n}$ is the unit normal vector of S_i, directed outwards from cell i, and dA is the surface element. If $\mathbf{F}$ is assumed to be constant over each face of cell i, equation 2.12 may be rearranged to give

$$\frac{\partial \phi}{\partial t} \delta V_i + A_{i+\frac{1}{2}} F\left(\phi_{i+\frac{1}{2}}\right) - A_{i-\frac{1}{2}} F\left(\phi_{i-\frac{1}{2}}\right) = 0, \tag{2.13}$$

where $F = \mathbf{F} \cdot \mathbf{e}_x$, and $A_{i\pm\frac{1}{2}}$ is the area of the cell face at $x_{i\pm\frac{1}{2}}$. By discretising in space and time, and using the time integration scheme given by equation 2.4, the standard first-order finite volume equation can be obtained. In one dimension, this is

$$\phi_i^{n+1} = \phi_i^n - \frac{\delta t}{\delta V_i} \left(A_{i+\frac{1}{2}} \hat{F}^n_{i+\frac{1}{2}} - A_{i-\frac{1}{2}} \hat{F}^n_{i-\frac{1}{2}} \right), \tag{2.14}$$

where $\hat{F}^n_{i\pm\frac{1}{2}}$ is the numerical flux at $x_{i\pm\frac{1}{2}}$, an approximation to $F\left(\phi^n_{i\pm\frac{1}{2}}\right)$. As before, the value of δt is set by the Courant condition.

The method of solution for a finite volume scheme is as follows. At time t the solution is known, and is approximated by the set of discrete values ϕ^n_i, $i \in [1, m]$. For every cell i, (1) a reconstruction procedure is used to calculate cell edge properties $\phi^n_{i\pm\frac{1}{2}}$; (2) numerical fluxes $\hat{F}^n_{i\pm\frac{1}{2}}$, are calculated from the cell edge properties and some numerical flux function, and (3) the numerical fluxes are used with equation 2.14 to obtain the solution at $t + \delta t$, ϕ^{n+1}_i. In some cases, step (1) is implicitly contained within step (2).

2.5 Reconstruction procedures

The are many possible methods of reconstructing cell edge properties from cell centred values. The most simple is to assume that properties are piecewise constant within each cell, as Godunov (1959) does, and hence set $\phi^n_{i\pm\frac{1}{2}} = \phi^n_i$. Piecewise constant reconstruction leads to a scheme which is first-order accurate in space. This reconstruction procedure is illustrated in the top left panel of Figure 2.4. An improvement to piecewise constant reconstruction is to assume that ϕ varies linearly within each computational cell. The Monotone Upstream-Centred Scheme for Conservation Laws (MUSCL) of van Leer (1979) assumes linear variation of properties within each cell, and uses slope limiters to prevent spurious oscillations from occurring near shocks and discontinuities. MUSCL schemes are second-order accurate in space. The MUSCL reconstruction procedure is shown in the top right panel of Figure 2.4. Higher order reconstruction procedures exist, such as the third-order Piecewise Parabolic Method (PPM) due to Colella and Woodward (1984) (illustrated in the bottom panel of Figure 2.4), in which properties are assumed to obey parabolic profiles within each cell, and the ENO and WENO schemes mentioned in Section 2.2, where polynomial interpolations are used to estimate cell edge properties.

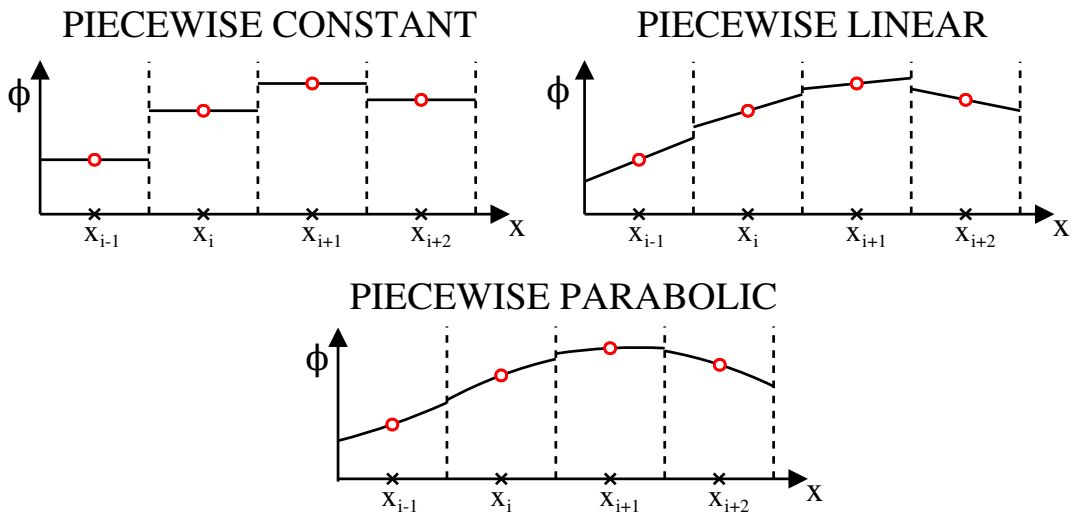


Figure 2.4: Different reconstruction procedures for finite volume schemes.

2.6 Riemann problems

The Riemann problem is an initial value problem for conservation laws, and is a useful problem for understanding the Euler equations. The solution to a Riemann problem is a similarity solution, and all properties of the solution are described by characteristics. Whilst the Riemann problem is a theoretical problem, the use of Riemann problems arises naturally in finite volume schemes, where they may be used to calculate fluxes at cell faces. As it is possible to obtain the exact solution to a Riemann problem, they are often used to provide simple test problems against which to validate numerical schemes. Whilst a Riemann problem may be constructed for any conservation law or set of conservation laws, for simplicity I limit the detail here to the specific case of the Euler equations.

The Euler equations in Cartesian coordinates and one spatial dimension are

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0, \tag{2.15}$$

where $\mathbf{U} = [\rho,\, \rho u,\, E]^T$ and $\mathbf{F} = \mathbf{F}(\mathbf{U}) = \left[\rho u,\, \rho u^2 + p,\, u\,(E+p)\right]^T$, in which ρ, u, p and E are the density, velocity, pressure and total energy. Equation 2.15 is closed with a general equation of state of the form $p = p\,(\rho,\, e)$, where the specific energy e is related to the total energy E by $E = \rho e + \frac{1}{2}\rho u^2$.

The problem consists of piecewise constant initial data containing a single discontinuity, such that at $t = 0$

$$\mathbf{U}(x, t = 0) = \begin{cases} \mathbf{U}_L & x \leq 0 \\ \mathbf{U}_R & x > 0. \end{cases} \tag{2.16}$$

The fluids on either side of the interface may or may not obey different equations of state. A Riemann problem is often denoted by $R(\mathbf{U}_L, \mathbf{U}_R)$.

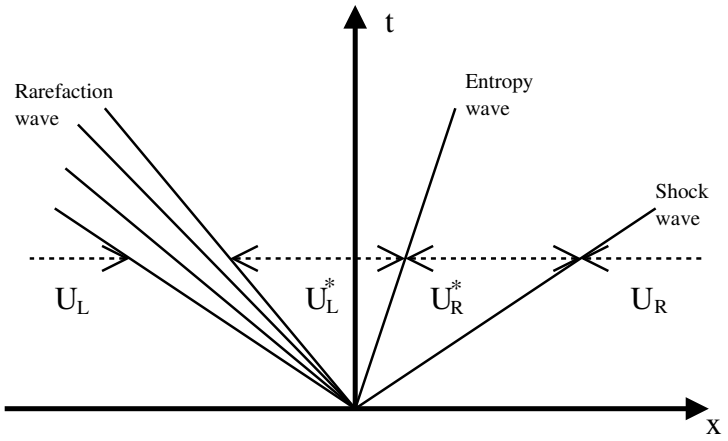


Figure 2.5: Solution space for a typical Riemann problem for the Euler equations in the $x - t$ plane. The abscissa and ordinate meet at $x = t = 0$.

Figure 2.5 shows the characteristics of a typical solution to a Riemann problem. Note that the characteristics trace straight lines in the x-t plane, as the solution is a function of x/t only. The solution contains three characteristic waves: the inner is an entropy wave (or contact discontinuity) and the outer two may be either shock waves or rarefaction fans. The properties in the regions between the waves are uniform and are called the star states, denoted $\mathbf{U}_L^\star$ and $\mathbf{U}_R^\star$. In certain cases some of the waves may have an amplitude of zero. Note that whilst in Figure 2.5 the solution contains a left-going rarefaction wave and a right-going shock wave, the directions of the shock and rarefaction may be reversed, or indeed, if the motion in the star states is supersonic all three waves will propagate in the same direction. It is also possible for the solution to contain two shocks, or two rarefaction fans. The Rankine-Hugoniot conditions constrain the pressure and velocity across the contact discontinuity to be continuous, such that $u_L^\star = u_R^\star = u^\star$ and $p_L^\star = p_R^\star = p^\star$, whilst $\rho_L^\star$ does not necessarily equal $\rho_R^\star$.

One of the most commonly used Riemann problems in testing numerical schemes is the Sod shock tube problem (Sod, 1978). The initial conditions for this problem are given by

$$(\rho,\, u,\, p,\, \gamma) = \begin{cases} (1,\, 0,\, 1,\, 1.4) & \text{if } x \leq 0, \\ (0.125,\, 0,\, 0.1,\, 5/3) & \text{if } x > 0, \end{cases} \tag{2.17}$$

where γ is the ratio of specific heats, and is used in the ideal gas equation of state $p = (\gamma - 1)\,\rho e$. Figure 2.6 shows the exact solution, calculated at a resolution of 10^{-3}, and the solution obtained from a simple first-order Godunov-type (Godunov, 1959) finite volume scheme with 200 cells across the computational domain. The solution is shown at $t = 0.2$. At this time, the rarefaction wave extends between approximately $x = -0.27$ and $x = -0.02$. The contact discontinuity is at $x = 0.17$ and the shock wave is at $x = 0.38$. The exact solution shows a sharp discontinuity at the shock wave and contact discontinuity, whilst the numerical solution has smeared out all three waves due to the numerical viscosity inherent in the scheme.

It is possible to calculate an exact solution to a Riemann problem, although for the Euler equations this requires an iterative procedure. Using the Rankine-Hugoniot conditions for shock waves, and considerations of entropy and invariancy along characteristics for rarefaction waves, relations between the states either side of the outer waves are obtained. The Rankine-Hugoniot conditions then constrain the pressure and velocity across the contact discontinuity to be continuous, allowing the expressions for the outer waves to be related to each other, leading to a single non-linear equation for one of the star state properties (eg. $p^\star$). This single equation is solved iteratively, after which the remaining star properties are calculated from the relations between the states across the outer waves.

Algorithms for the exact solution of Riemann problems are presented in, for example, Glimm (1965), Chorin (1976), Colella and Woodward (1984) and Colella (1985). Ivings *et al.* (1998) provides details of exact Riemann solvers for the solution of Riemann problems with liquid equations of state. Due to the iterative nature of exact Riemann solvers, they are too computationally expensive to be of use in many finite volume schemes. Approximate Riemann solvers are computationally cheaper and often more

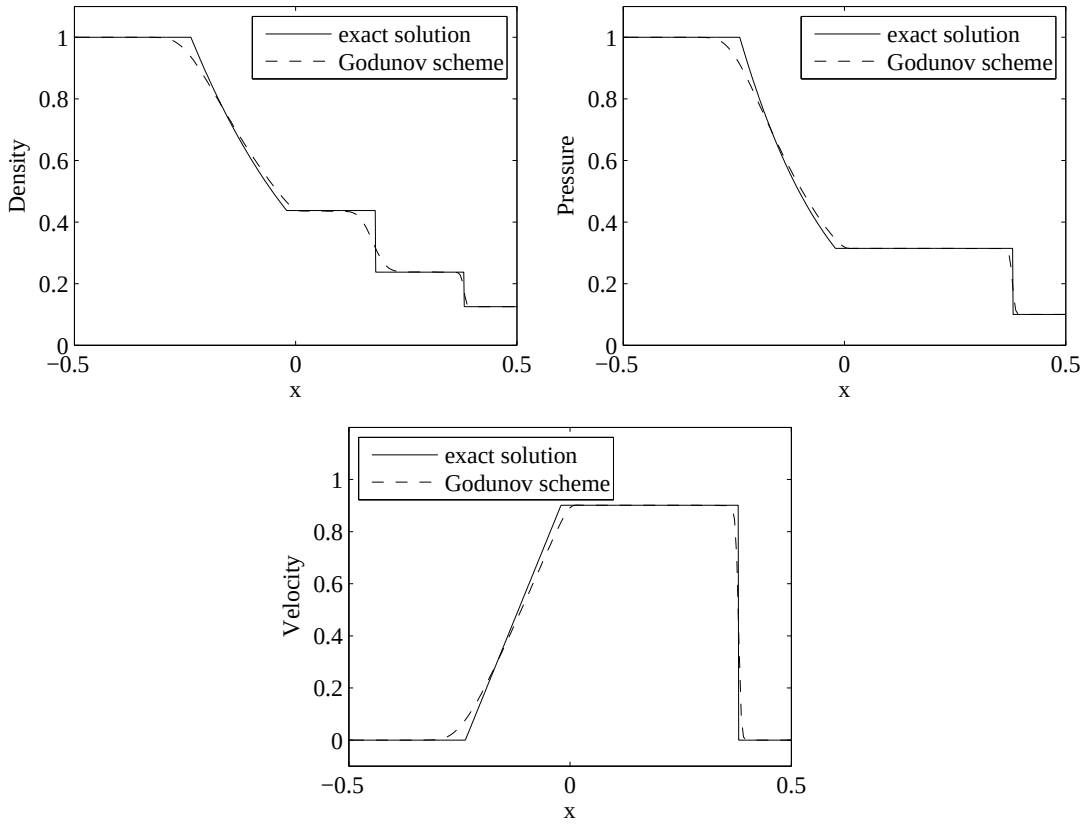


Figure 2.6: Spatial profiles of density, pressure and velocity for the Sod shock tube problem at $t = 0.2$. Plots show the exact solution (solid line) and the solution obtained with a basic finite volume scheme (dashed-line).

robust, as some of the iterative methods used in exact Riemann solvers may fail to converge at all. Approximate Riemann solvers have been developed by, for example, Harten and Lax (1981), Harten *et al.* (1983) and Roe (1986). Ivings *et al.* (1998) present several approximate Riemann solvers for liquids. Hu *et al.* (2009) describe an approximate Riemann solver, based on a modified version of the averaging procedure of Roe (1986), capable of solving for different fluids with a general equation of state. I describe the approximate Riemann solver of Hu *et al.* (2009) in Appendix A.

2.7 Numerical fluxes

Early finite volume schemes, such as that of Godunov (1959) or Glimm (1965) defined Riemann problems at cell interfaces, but did not use the solution to obtain fluxes, instead using the solution to the Riemann problems to obtain directly the solution at the next time step. Godunov's scheme (Godunov, 1959) takes the solution at the subsequent time step as the average solution to the Riemann problems defined from the solution at the current time step. The random choice method (RCM) of Glimm (1965) takes the solution at the subsequent time step as the solution to the Riemann problems at a random point in each cell. Other methods were developed, which calculate fluxes without the use of Riemann solvers, such as Lax-Friedrichs (LF) flux and local Lax-Friedrichs (LLF) fluxes (Lax, 1954). These are simple, computationally cheap to implement, and provide very smooth fluxes through the introduction of artificial viscosity terms in the fluxes. The Lax-Friedrichs flux is given by (Lax, 1954)

$$\hat{F}^n_{i+\frac{1}{2}} = f^+_i + f^-_{i+1}, \tag{2.18}$$

where

$$f^+ = \frac{1}{2}\left(F\left(\phi^n_i\right) + \alpha\phi^n_i\right), \tag{2.19a}$$

$$f^- = \frac{1}{2}\left(F\left(\phi^n_i\right) - \alpha\phi^n_i\right), \tag{2.19b}$$

where $\alpha \geq \max\left|F'\left(\phi\right)\right|$. The Lax-Friedrichs flux is often used as a low order building block for high order schemes. Note that if equation 2.10 were solving using piecewise constant reconstruction, the Lax-Friedrichs flux and first-order time integration, the resulting scheme would be identical to the first-order upwind finite difference scheme described in Section 2.2. Later, various numerical fluxes, which use some form of characteristic decomposition or Riemann problem to determine fluxes between cells were developed (for example, Harten *et al.* (1983), Colella and Woodward (1984), Colella (1985), Roe (1986), Einfeldt (1988) and Toro *et al.* (1994)). Of these, the HLLC flux of Toro *et al.* (1994) is perhaps most widely used. It should be noted that whilst the scheme of Godunov (1959) did not explicitly calculate fluxes, the terms 'Godunov scheme' and 'Godunov-type scheme' are often used to describe any scheme in which

fluxes are obtained from the solutions to Riemann problems at cell faces. Another class of schemes, based on the concept of flux splitting, has achieved popularity. These schemes fall into two types: (1) flux difference splitting, a well known example of which is the scheme of Roe (1997), and (2) flux vector splitting, where the scheme of van Leer (1982) is a good example. Another example of flux vector splitting schemes are advective upstream splitting methods (AUSM), which were developed by Liou and Steffen (1993); Liou (1996). AUSM separates the fluxes into convective fluxes and pressure fluxes, where convective fluxes advect with the local velocity, and pressure fluxes advect at the local speed of sound. This approach allows for non-Cartesian coordinate systems to be accounted for within the fluxes, and is gaining popularity, especially in schemes with adaptive mesh refinement, and arbitrary Lagrangian-Eulerian schemes.

2.8 Time integration schemes

The time-integration method used in equation 2.14 is the forward Euler method, which is first-order accurate in time. In some cases, the forward Euler method does not provide sufficient accuracy, and there has been much work on higher order methods. I define

$$L_i^n = -\frac{1}{\delta V_i}\left(A_{i+\frac{1}{2}}\hat{F}^n_{i+\frac{1}{2}} - A_{i-\frac{1}{2}}\hat{F}^n_{i-\frac{1}{2}}\right), \tag{2.20}$$

allowing equation 2.14 to be expressed as

$$\phi^{n+1} = \phi^n + \delta t L^n, \tag{2.21}$$

where the subscript i has been dropped for ease of exposition. A second-order time integration method is the mid-point method, given by

$$\phi^{(1)} = \phi^n + \frac{\delta t}{2}L^n \tag{2.22a}$$

$$\phi^{n+1} = \phi^n + \frac{\delta t}{2}L^{(1)} \tag{2.22b}$$

where a superscript in parentheses denotes an intermediate step. The Lax-Wendroff method (Lax and Wendroff, 1960) is another two-step time integration method, which is second-order accurate. Properties at the half time step $n + \frac{1}{2}$ are calculated on a

staggered grid (at points $x_{i\pm\frac{1}{2}}$) in the first step. In the second step, these properties are used to obtain the solution at $n+1$. The Lax-Wendroff method is comparatively complex to program. A more widely used type of time integration is Runge-Kutta methods. Runge-Kutta methods are simpler to program than Lax-Wendroff methods, especially for multi-dimensional problems and problems with source terms. Runge-Kutte methods involve making several steps of differing sizes forward in time, then using a weighted average of these steps to cancel out low order error terms. The first-order forward Euler method described by equation 2.21, and the second-order midpoint method described by equation 2.22, are types of Runge-Kutta method. The most widely used Runge-Kutta method is the classic fourth-order scheme, described by (Robinson (2004) for example)

$$\phi^{(1)} = \phi^n + \frac{\delta t}{2} L^n \tag{2.23a}$$

$$\phi^{(2)} = \phi^n + \frac{\delta t}{2} L^{(1)} \tag{2.23b}$$

$$\phi^{(3)} = \phi^n + \delta t L^{(2)} \tag{2.23c}$$

$$\phi^{n+1} = \phi^n + \frac{\delta t}{6}\left[L^n + 2L^{(1)} + 2L^{(2)} + L^{(3)}\right] \tag{2.23d}$$

Shu and Osher (1988) investigated high-order TVD time integration schemes which allowed for higher CFL numbers than traditional Runge-Kutta methods. They used an ENO reconstruction procedure, with the Lax-Freidrichs flux as the basic building block. A second-order TVD scheme due to Shu and Osher (1988) is

$$\phi^{(1)} = \phi^n + \delta t L^n \tag{2.24a}$$

$$\phi^{n+1} = \frac{1}{2}\phi^n + \frac{1}{2}\phi^{(1)} + \frac{1}{2}\delta t L^{(1)}, \tag{2.24b}$$

and their third-order scheme is

$$\phi^{(1)} = \phi^n + \delta t L^n \tag{2.25a}$$

$$\phi^{(2)} = \frac{3}{4}\phi^n + \frac{1}{4}\phi^{(1)} + \frac{1}{4}\delta t L^{(1)} \tag{2.25b}$$

$$\phi^{n+1} = \frac{1}{3}\phi^n + \frac{2}{3}\phi^{(2)} + \frac{2}{3}\delta t L^{(2)}. \tag{2.25c}$$

2.9 Irregular and moving meshes

Throughout this thesis I solve equations on a fixed computational mesh, which does not move, and is not refined during the computation. However, I frequently refer to other works in which this is not the case, so I introduce the concepts of moving meshes and mesh refinement here.

In this chapter I consider solutions to the Euler equations on a computational mesh which is fixed in space. Numerical schemes of this form are termed *Eulerian*. However, the governing equations may be recast to create a scheme in which the computational mesh moves with the fluid. This type of scheme is referred to as *Lagrangian*. A generalisation of moving mesh-schemes are *arbitrary Lagrangian-Eulerian* (ALE) schemes, in which the motion of the mesh is arbitrary. In some regions it may be stationary, and in others may move with the fluid velocity, or even at the speed of sound. Lagrangian and ALE schemes are sometimes used to avoid difficulties with open boundary problems, by dynamically expanding the domain to contain the entire solution.

When numerically solving hyperbolic PDEs, the solution obtained frequently contains a large range of length-scales. Some regions of the solution may contain very little detail, whilst others contain very fine detail. If the mesh is too coarse, the simulation may fail to capture important detail which is crucial to understand the implications of the solution. However, it is computationally inefficient to conduct the entire simulation at a resolution capable of capturing the finest details desired. This apparent conflict between computational cost and resolution of results can be partially mitigated by using an irregular mesh. The mesh may be coarse in regions where little detail is required, and finer in those regions of interest. However, the regions of interest are not always stationary, and often move through the solution. In this case, the mesh may be dynamically refined in those regions of interest as the simulation progresses. This approach is called *adaptive mesh refinement* (AMR), and is common in the field of computational fluid dynamics.

Chapter 3

Artificial boundary conditions using the NLAA

3.1 Introduction

Many problems in fluid dynamics are posed in unbounded domains. Various methods are employed to enable these problems to be solved numerically, a common one of which is to restrict the computation to a finite region and impose artificial boundary conditions on the truncated domain. The aim of the artificial boundary conditions is to mimic the unbounded domain and prevent spurious reflections from the domain boundary. Artificial boundary conditions of this type are often referred to as 'non-reflecting' and 'absorbing'. Inaccurate artificial boundary conditions can lead to spurious disturbances at the domain boundaries, which propagate back through the domain, contaminating results.

There has been much work on non-reflecting boundary conditions. Reviews are given by Givoli (1991), Hagstrom (1999) and Tsynkov (1998). Many methods have been derived for wave propagation problems, such as the perfectly matched layer method (Berenger, 1994, 1996) which are only strictly applicable to linear hyperbolic systems. However

similar methods for the Euler equations have been developed (Hayder *et al.*, 1999). A thorough review of work on artificial boundary conditions for compressible flow is given by Colonius (2004).

Hedstrom (1979) decomposed the Euler equations into characterstic wave equations and presented a non-reflecting boundary condition using these equations. Thompson (1990) developed a useful formalism for applying characteristic boundary conditions, and described a method of applying non-reflecting characteristic boundary conditions (Thompson, 1987). The characteristic boundary condition formalism of Thompson (1990) is widely used.

The behaviour of oscillating air bubbles in water is of interest in a variety of fields, including cavitation, underwater explosions and shock wave lithotripsy (a medical procedure to break down kidney stones into crystals small enough to be passed out in the urine). Rayleigh (1917) developed an equation of motion for a spherical cavity in an infinite incompressible fluid. The analysis in Rayleigh (1917) forms the basis for much work on oscillating bubbles. Lamb (1923) derived an exact wave equation for the motion of a spherical cavity in a compressible fluid, and obtained an analytical solution for the special case of the ratio of specific heats, γ, being equal to $4/3$. In general there is no analytical solution to this equation. Extensions to Rayleigh (1917) accounting for the compressibility of water were developed by several authors (Herring (1941) and Keller and Kolodner (1956)). Gilmore (1952) developed the work of Rayleigh (1917), Lamb (1923) and Herring (1941) to obtain an algorithm to calculate the propagating wavefield outside the bubble. This scheme requires further approximations, and is less accurate than the equations of motion on which it is based. Although on a much smaller scale, cavitation bubbles are a physically similar phenomenon, and similar approximations have been developed to describe cavitation bubbles (Plesset (1949), Plesset and Zwick (1952) and Prosperetti and Plesset (1978)), based on the widely used Rayleigh-Plesset equation.

Marine seismic exploration can be thought of as a powerful form of echo sounding, capable of penetrating the sea floor, to enable a three-dimensional image of the sub-sea

to be created. It is a process used in the petroleum industry in the search for geological features which have the potential to contain trapped hydrocarbons. Initially dynamite was used as the source in marine seismic exploration. However environmental concerns led to the development of alternative sources. Currently, seismic air guns are the most commonly used source. In use they are towed behind a ship, usually between 5 and 20 metres beneath the sea surface, and when 'fired' release a quantity of air at high pressure (136 atm), that forms a bubble which oscillates, producing a wavefield which propagates through the sea and into the subsurface. A seismic air gun is analagous to a weak underwater explosion.

Air gun bubbles were first modelled in 1970 by Ziolkowski (1970), using a simplified two-equation ordinary differential equation model of a seismic air gun based on the work of Gilmore (1952). This method is still the basis of the modelling currently used by industry. The non-linear acoustic approximation (NLAA) was developed by Ziolkowski (1998) as an improvement to Ziolkowski (1970), and is equivalent to the approximations of Herring (1941) and Keller and Kolodner (1956). The NLAA approximates the wavefield produced by an oscillating bubble - subject to certain assumptions - and allows the calculation of pressure and velocity at any point provided the pressure and velocity are known at a single location. Boundary integral methods which allowed the simulation of non-spherical bubbles have been developed for cavitation modelling (Kucera and Blake (1990), Hooton *et al.* (1994) and Blake *et al.* (1999)). Cox *et al.* (2004) provide a good review of air gun modelling, and apply earlier boundary integral methods to seismic air guns.

The first finite volume simulations of underwater explosions appear in Flores and Holt (1981), although a lack of adequate boundary conditions means that only very early stages of the explosion were calculated. A review of early work on underwater explosions is provided in Holt (1977). The one-dimensional spherically symmetric underwater explosion has since become a commonly used test case (although no analytical solution exists) for multimedium Euler solvers, and has been simulated using a variety of numerical schemes (Cocchi *et al.* (1996), Wardlaw and Mair (1998), Smith (1999), Liu *et al.* (2001a), Hu *et al.* (2006), Pishevar and Amirifar (2010) and Barras *et al.*

(2012)). In a typical underwater explosion problem the outgoing pressure wavefield propagates to approximately 100 times the maximum bubble radius during a single bubble oscillation. Previous simulations have relied either on the use of a very large domain (for example Hu *et al.* (2006)) which is computationally expensive, or on arbitrary Lagrangian-Eulerian (ALE) methods (Pishevar and Amirifar (2010), Barras *et al.* (2012) and Smith (1999)) in which the problem is solved on a mesh which expands to contain the outgoing wavefield.

My solution is to take the non-linear acoustic approximation and use it to develop artificial boundary conditions for a finite volume simulation of an oscillating bubble on a truncated domain. I base my approximation on the conditions at the domain boundary. This approximation is then used to describe any incoming characteristic waves. These characteristic waves are then applied through the characteristic boundary condition formalism of Thompson (1990). This method allows finite volume simulations of oscillating bubbles to be carried out for long run-times on comparatively small domains, thus reducing computational costs.

The layout of this chapter is as follows. In Section 3.2 I show the derivation of the NLAA. In Section 3.3 I present a brief summary of the characteristic boundary condition formalism. In Section 3.4 I use the NLAA to derive artificial boundary conditions. In Section 3.5 I discuss the range of validity of the NLAA. Section 3.6 is a summary.

3.2 The non-linear acoustic approximation

Ziolkowski (1998) developed the non-linear acoustic approximation for the motion of a spherical bubble in water for use in modelling seismic air guns. The approximation is based on the assumption that the acoustic wavefield produced by the bubble is dominated by wavelengths many times the bubble diameter, which allows the bubble to be considered a monopole source. The velocity is described by a velocity potential which is assumed to obey the linear acoustic wave equation, leading to an analytical solution for the velocity potential. This solution is then passed back into the Euler equations

to obtain solutions for the pressure and velocity. The following is the derivation from Ziolkowski (1998).

Starting from Lamb (1923), the bubble is assumed to be spherical, and all motion is in the radial direction and subject to spherical symmetry. The local specific enthalpy in the water, h, is defined as

$$h = \int_{p_\infty}^{p} \frac{dp}{\rho} = \int_{\rho_\infty}^{\rho} c^2 \frac{d\rho}{\rho}, \tag{3.1}$$

where p is the pressure, ρ is the density, and c is the speed of sound, defined by $c^2 = dp/d\rho|_{isentropic}$ and p_∞ and ρ_∞ are the pressure and density in the undisturbed water. For the pressure fluctuations considered, it is acceptable to assume that $\rho = \rho_\infty$ and $h = (p - p_\infty)/\rho_\infty$. The speed of sound in the water is assumed to be constant. It should be noted that this combination of assumptions - incompressible flow and finite speed of sound - is a contradictory set of assumptions, but acceptable because of the low Mach number flows involved. The fluid can be considered incompressible because the variation in density is so small. However, according to the governing equations, in the limit of incompressibility, the speed of sound tends to infinity. The approximation here is not that the speed of sound remains finite - it clearly does, and it is important to note arguments of causality which, if ignored, could lead to the presumption that the speed of sound increases as the signals decrease in amplitude. The approximation is that the density remains constant. Using Bernoulli's equation, and a value for the compressibility of water of $5.1 \times 10^{-10}\text{Pa}^{-1}$ (Fine and Millero, 1973), it can easily be shown that the density variation of water for flow with a Mach number of 0.1 is approximately 0.6%. On these grounds, this assumption seems reasonable. Further discussion is given at the end of this chapter. Viscosity is neglected (Ziolkowski, 1970). The flow is assumed to be irrotational and the velocity $\mathbf{u} = u\mathbf{e_r}$ obeys a velocity potential such that $\mathbf{u} = -\nabla\phi$. Hence $u = -\partial\phi/\partial r$. The equation of motion is written

$$\frac{D\mathbf{u}}{Dt} + \nabla h = 0, \tag{3.2}$$

where $D(\cdot)/Dt$ is the material derivative, defined by $D(\cdot)/Dt = \partial(\cdot)/\partial t + u\nabla(\cdot)$. Equation 3.2 is integrated to give

$$h = \frac{\partial\phi}{\partial t} - \frac{u^2}{2}. \tag{3.3}$$

An equation for the conservation of mass is written

$$\frac{1}{\rho}\frac{D\rho}{Dt} - \nabla^2\phi = 0. \tag{3.4}$$

Equations 3.1 and 3.4 are combined, yielding

$$\frac{Dh}{Dt} = \frac{c^2}{\rho}\frac{D\rho}{Dt} = c^2\nabla^2\phi. \tag{3.5}$$

With the imposition of spherical symmetry, from equations 3.3 and 3.5 the exact wave equation, first derived by Lamb (1923), is obtained

$$\frac{\partial^2\phi}{\partial r^2}\left(1 - \frac{u^2}{c^2}\right) + \frac{2}{r}\frac{\partial\phi}{\partial r}\left(1 + \frac{r}{c^2}\frac{\partial^2\phi}{\partial t\partial r}\right) - \frac{1}{c^2}\frac{\partial^2\phi}{\partial t^2} = 0. \tag{3.6}$$

Equation 3.6 has no known analytical solution in general. If the advective terms - $u\partial\left(\cdot\right)/\partial r$ - in equations 3.2, 3.4 and 3.5 are neglected, equation 3.6 collapses to the linear acoustic wave equation

$$\frac{\partial^2\phi}{\partial r^2} + \frac{2}{r}\frac{\partial\phi}{\partial r} - \frac{1}{c^2}\frac{\partial^2\phi}{\partial t^2} = 0. \tag{3.7}$$

Lamb (1923) estimated that for flows with Mach number less than approximately 0.1, the errors caused by this approximation would be less than 1%. The wavelengths of the pressure field produced by the bubble are large compared with the bubble radius, hence the bubble can be considered a point source. There are no other sources. Under these conditions, equation 3.7 has the well known solution

$$\phi\left(r,t\right) = \frac{1}{r}f\left(t - \frac{r}{c}\right). \tag{3.8}$$

Differentiation of equation 3.8 yields

$$u\left(r,t\right) = -\frac{\partial\phi}{\partial r} = \frac{1}{r^2}f + \frac{1}{rc}f' \tag{3.9}$$

where the argument of f, $(t - r/c)$, has been dropped for ease of writing, and a prime denotes differentiation with respect to the argument. Further differentiation gives

$$\frac{\partial u}{\partial r} = -\frac{\partial^2\phi}{\partial r^2} = \frac{-2}{r^3}f - \frac{2}{r^2c}f' - \frac{1}{rc^2}f'' \tag{3.10}$$

and

$$\frac{\partial u}{\partial t} = -\frac{\partial^2\phi}{\partial r\partial t} = \frac{1}{r^2}f' + \frac{1}{rc}f''. \tag{3.11}$$

These results are passed into equations 3.2 and 3.4 giving

$$\frac{1}{r^2}f' + \frac{1}{rc}f'' + \frac{1}{\rho}\frac{\partial p}{\partial r} + u\left(\frac{-2}{r^3}f - \frac{2}{r^2c}f' - \frac{1}{rc^2}f''\right) = 0. \tag{3.12}$$

Equations 3.3, 3.8, 3.9 and 3.11 are used to provide expressions for f, f' and f''

$$f = r^2\left(u - \frac{h}{c} - \frac{u^2}{2c}\right), \tag{3.13}$$

$$f' = r\frac{\partial \phi}{\partial t} = r\left(h + \frac{u^2}{2}\right), \tag{3.14}$$

$$f'' = rc\frac{\partial u}{\partial t} - ch - \frac{cu^2}{2}. \tag{3.15}$$

Ziolkowski (1998) argued that the quantity $r\left(h + u^2/2\right)$ propagates outwards at speed c with uniform amplitude, and determined a Lagrangian form of this result, obtaining

$$R\ddot{R}\left(1 - \frac{2\dot{R}}{c}\right) + \frac{3\dot{R}^2}{2}\left(1 - \frac{4\dot{R}}{3c}\right) = H + \frac{R\dot{H}}{c}\left(1 - \frac{\dot{R}}{c}\right), \tag{3.16}$$

where R is the bubble radius, H the enthalpy of the water at the bubble wall and c the speed of sound of water. A dot represents differentiation with respect to time. This result is solved numerically along with an equation such as $PR^{3n} = constant$, where n is a constant and $1 \leq n \leq 1.4$, to simulate the evolution of the bubble through time. I give details of air gun bubble models of this form in Appendix D. P and R are also used with the above results to calculate an approximation of the pressure and velocity at any point in the water.

3.3 Characteristic boundary condition formalism

Consider a finite spherical domain Ω, of radius R_D, bounded by Γ. Thompson (1990) presents a formalism for the treatment of boundary conditions in finite difference simulations for hyperbolic systems of conservation laws. This method decomposes the system of equations into a set of uncoupled wave equations for non-linear characteristics. This set of equations is then solved on domain boundaries, with any incoming characteristic waves being specified according to the boundary condition desired. The following derivation follows Thompson (1990), but using a polar coordinate system.

In polar coordinates in two dimensions, the Euler equations may be written in primitive form as

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial r} + \frac{v}{r}\frac{\partial \rho}{\partial \theta} + \rho\left(\frac{\partial u}{\partial r} + \frac{1}{r}\frac{\partial v}{\partial \theta}\right) + \frac{2\rho u}{r} + \frac{\rho v}{r\tan\theta} = 0, \tag{3.17}$$

$$\frac{\partial p}{\partial t} + u\frac{\partial p}{\partial r} + \frac{v}{r}\frac{\partial p}{\partial \theta} + \rho c^2\left(\frac{\partial u}{\partial r} + \frac{1}{r}\frac{\partial v}{\partial \theta}\right) + \frac{2\rho c^2 u}{r} + \frac{\rho v c^2}{r\tan\theta} = 0, \tag{3.18}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial r} + \frac{v}{r}\frac{\partial u}{\partial \theta} + \frac{1}{\rho}\frac{\partial p}{\partial r} + \frac{v^2}{r} + g\cos\theta = 0, \tag{3.19}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial r} + \frac{v}{r}\frac{\partial v}{\partial \theta} + \frac{1}{r\rho}\frac{\partial p}{\partial \theta} + \frac{uv}{r} - g\sin\theta = 0, \tag{3.20}$$

where ρ, p, u and v are the density, pressure, radial velocity and polar velocity. c is the speed of sound, and g is the acceleration due to gravity. Equations 3.17 to 3.20 can be expressed as

$$\frac{\partial \tilde{\mathbf{U}}}{\partial t} + \mathbf{A}\frac{\partial \tilde{\mathbf{U}}}{\partial r} + \mathbf{C} = 0 \tag{3.21}$$

where $\tilde{\mathbf{U}} = [\rho,\, p,\, u,\, v]$,

$$\mathbf{A} = \begin{bmatrix} u & 0 & \rho & 0 \\ 0 & u & \rho c^2 & 0 \\ 0 & 1/\rho & u & 0 \\ 0 & 0 & 0 & u \end{bmatrix} \tag{3.22}$$

and

$$\mathbf{C} = \begin{bmatrix} \frac{v}{r}\frac{\partial \rho}{\partial \theta} + \frac{\rho}{r}\frac{\partial v}{\partial \theta} + \frac{2\rho u}{r} + \frac{\rho v}{r\tan\theta} \\ \frac{v}{r}\frac{\partial p}{\partial \theta} + \frac{\rho c^2}{r}\frac{\partial v}{\partial \theta} + \frac{2\rho c^2 u}{r} + \frac{\rho v c^2}{r\tan\theta} \\ \frac{v}{r}\frac{\partial u}{\partial \theta} + \frac{v^2}{r} + g\cos\theta \\ \frac{v}{r}\frac{\partial v}{\partial \theta} + \frac{1}{r\rho}\frac{\partial p}{\partial r} + \frac{uv}{r} - g\sin\theta \end{bmatrix} \tag{3.23}$$

is a vector of source terms and θ-derivatives. The eigenvalues λ of $\mathbf{A}$ can be calculated by solving

$$\det\left(\mathbf{A} - \lambda\mathbf{I}\right) = 0, \tag{3.24}$$

where $\mathbf{I}$ is the identity matrix, obtaining

$$\lambda_1 = u - c, \quad \lambda_2 = \lambda_3 = u, \quad \lambda_4 = u + c. \tag{3.25}$$

λ_1 and λ_4 are the velocities of left and right moving sound waves. λ_2 is the velocity at which entropy is advected, and λ_3 is the advection velocity of v. The left eigenvectors

$\mathbf{l}_i^T$ of $\mathbf{A}$ are then found to be

$$\mathbf{l}_1^T = (0,\, 1,\, -\rho c,\, 0)\,, \tag{3.26a}$$

$$\mathbf{l}_2^T = \left(c^2,\, -1,\, 0,\, 0\right), \tag{3.26b}$$

$$\mathbf{l}_3^T = (0,\, 0,\, 0,\, 1)\,, \tag{3.26c}$$

$$\mathbf{l}_4^T = (0,\, 1,\, \rho c,\, 0)\,. \tag{3.26d}$$

A matrix $\mathbf{S}$ is formed, where the columns of $\mathbf{S}$ are the right eigenvectors of $\mathbf{A}$ and the rows of $\mathbf{S}^{-1}$ are the left eigenvectors, such that $\mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1} = \mathbf{A}$, where $\mathbf{\Lambda}$ is a diagonal matrix of the eigenvalues of $\mathbf{A}$. Equation 3.21 may now be written

$$\frac{\partial \tilde{\mathbf{U}}}{\partial t} + \mathbf{S}\mathbf{L} + \mathbf{C} = 0, \tag{3.27}$$

where the components of $\mathbf{L}$, $\mathscr{L}_i$, are defined by

$$\mathscr{L}_i \equiv \lambda_i \mathbf{l}_i^T \frac{\partial \tilde{\mathbf{U}}}{\partial r}. \tag{3.28}$$

The $\mathscr{L}_i$ are defined as

$$\mathscr{L}_1 = \lambda_1 \left\{ \frac{\partial p}{\partial r} - \rho c \frac{\partial u}{\partial r} \right\}, \tag{3.29a}$$

$$\mathscr{L}_2 = \lambda_2 \left\{ c^2 \frac{\partial \rho}{\partial r} - \frac{\partial p}{\partial r} \right\}, \tag{3.29b}$$

$$\mathscr{L}_3 = \lambda_3 \left\{ \frac{\partial v}{\partial r} \right\}, \tag{3.29c}$$

$$\mathscr{L}_4 = \lambda_4 \left\{ \frac{\partial p}{\partial r} + \rho c \frac{\partial u}{\partial r} \right\}. \tag{3.29d}$$

The uncoupled wave equations for non-linear characteristics on Γ may now be written in full as

$$\frac{\partial \rho}{\partial t} + \frac{1}{c^2}\left\{ \mathscr{L}_2 + \frac{1}{2}\left[\mathscr{L}_4 + \mathscr{L}_1\right] \right\} + \frac{2\rho u}{r} + \frac{v}{r}\frac{\partial \rho}{\partial \theta} + \frac{\rho}{r}\frac{\partial v}{\partial \theta} + \frac{\rho v}{r \tan\theta} = 0, \tag{3.30}$$

$$\frac{\partial p}{\partial t} + \frac{1}{2}\left\{\mathscr{L}_4 + \mathscr{L}_1\right\} + \frac{2\rho c^2 u}{r} + \frac{v}{r}\frac{\partial p}{\partial \theta} + \frac{\rho c^2}{r}\frac{\partial v}{\partial \theta} + \frac{\rho v c^2}{r \tan\theta} = 0, \tag{3.31}$$

$$\frac{\partial u}{\partial t} + \frac{1}{2\rho c}\left\{\mathscr{L}_4 - \mathscr{L}_1\right\} + \frac{v}{r}\frac{\partial u}{\partial \theta} + \frac{v^2}{r} + g\cos\theta = 0, \tag{3.32}$$

$$\frac{\partial v}{\partial t} + \mathscr{L}_3 + \frac{v}{r}\frac{\partial v}{\partial \theta} + \frac{1}{r\rho}\frac{\partial p}{\partial \theta} + \frac{uv}{r} - g\sin\theta = 0, \tag{3.33}$$

where each of the four $\mathscr{L}_i$ describes a characteristic wave mode, each with propagation speed λ_i. The terms proportional to $2u/r$ and $1/r \tan\theta$ are source terms due to the polar coordinates.

Recall that finite difference simulations of the Euler equations involve the following scheme: (1) calculation of spatial derivatives based on the solution at the current time step; (2) using these spatial derivatives in some form of the Euler equations to determine time derivatives; then (3) integrating the time derivatives to obtain the solution at the next time step. On boundary nodes equations 3.30 to 3.33 are solved. At any point in space, each characteristic wave mode $\mathscr{L}_i$ is described entirely by information downstream of that point determined by the corresponding characteristic wave speed λ_i.

If the characteristic wave $\mathscr{L}_i$ is propagating into Ω on Γ ($\lambda_i < 0$ at $r = R_D$), the information describing that wave mode is contained entirely outside the domain, and hence $\mathscr{L}_i$ must be specified by some artificial boundary condition (for example, $\mathscr{L}_i = 0$). If the characteristic wave is propagating out of the domain then it is entirely defined by information contained within the domain, in which case equations 3.29 may be used to define $\mathscr{L}_i$, based upon the solution within the domain. For example, a zero-velocity boundary condition (a reflective boundary) on Γ is applied by computing $\mathscr{L}_4$ from its definition in equation 3.29d, prescribing $\mathscr{L}_1 = \mathscr{L}_4$ and $\mathscr{L}_2 = \mathscr{L}_3 = 0$, and then solving equations 3.30 to 3.33 at that point.

Thompson (1990) describes a 'non-reflecting' boundary condition as one in which all characteristic waves incoming to the domain are suppressed. To apply this boundary condition on Γ with subsonic flow, I compute $\mathscr{L}_4$ from its definition in equation 3.29d and prescribe $\mathscr{L}_1 = 0$. $\mathscr{L}_2 = \mathscr{L}_3 = 0$ if $u(b,t) \geq 0$, otherwise $\mathscr{L}_2$ and $\mathscr{L}_3$ are defined from equations 3.29b and 3.29c. I then solve equations 3.30 to 3.33 at that point. Thompson (1987) admits that there are many situations in which the correct solution does contain both outgoing and incoming characteristic waves, and demonstrates some of the limitations of this boundary condition.

3.4 Boundary conditions using the non-linear acoustic approximation

The application of artificial boundary conditions requires an approximation to the solution external to the domain. When using the formalism described in the previous section, spatial derivatives of the external solution are required. The NLAA allows the solution external to Ω to be estimated from the solution within Ω on Γ. From equations 3.8 to 3.15, spatial derivatives of the fluid properties on Γ due to the external solution can be calculated, based on the internal solution on Γ. Furthermore, these approximate spatial derivatives may be used to apply boundary conditions for the finite volume simulation. From equations 3.9 to 3.15 the spatial derivatives of the approximate solution external to Ω on Γ are related to the internal solution on Γ, according to

$$\left(\frac{\partial u}{\partial r}\right)_{NLAA} = \frac{u}{r}\left(\frac{u}{2c} - 2\right) + \frac{(p - p_\infty)}{\rho r c} - \frac{1}{c}\frac{\partial u}{\partial t}, \tag{3.34}$$

$$\left(\frac{\partial p}{\partial r}\right)_{NLAA} = \frac{\rho u^2}{r}\left(2 - \frac{u}{2c}\right) - \frac{u\,(p - p_\infty)}{rc} + \rho\left(\frac{u}{c} - 1\right)\frac{\partial u}{\partial t}, \tag{3.35}$$

$$\left(\frac{\partial \rho}{\partial r}\right)_{NLAA} = 0. \tag{3.36}$$

The NLAA is only valid for problems with spherical symmetry, in regions where the density variation is small, and velocities are small compared with sound speeds. Consider the domain Ω defined by the set of points $r \in [0, R_D]$. Within the domain there may be an air gun bubble, an underwater explosion, or some other source, but R_D is large enough that at $r = R_D$ the NLAA is valid. On R_D, $\lambda_1 < 0$ and $\lambda_4 > 0$. As such, $\mathscr{L}_1$ must be specified on the boundary from information based on the approximation to the exterior flow. The velocities in the water will sometimes be directed inwards and sometimes outwards. If $u\,(R_D, t) > 0$ then $\mathscr{L}_2$ and $\mathscr{L}_3$ can be calculated from equations 3.29b and 3.29c. If $u\,(R_D, t) \leq 0$ then $\mathscr{L}_2$ and $\mathscr{L}_3$ must be specified from information based on the external flow. $\mathscr{L}_4$ will always be set by equation 3.29d.

Prescription of $\mathscr{L}_1$

Equations 3.34 and 3.35 are passed back into the definition of $\mathscr{L}_1$ to obtain

$$\mathscr{L}_1 = \frac{\rho\,(u-c)}{r}\left\{\frac{u^2}{2}\left(3-\frac{u}{c}\right)+2uc-\frac{p-p_\infty}{\rho_\infty}\left(1+\frac{u}{c}\right)+\frac{u}{c}r\frac{\partial u}{\partial t}\right\}. \tag{3.37}$$

I write

$$\mathscr{L}_1 = \frac{\rho\,(u-c)\,u}{c}\frac{\partial u}{\partial t}+\alpha_{\mathscr{L}_1}, \tag{3.38}$$

where

$$\alpha_{\mathscr{L}_1} = \frac{\rho\,(u-c)}{r}\left\{\frac{u^2}{2}\left(3-\frac{u}{c}\right)+2uc-\frac{p-p_\infty}{\rho_\infty}\left(1+\frac{u}{c}\right)\right\}. \tag{3.39}$$

Equation 3.32 may now be expressed as

$$\frac{\partial u}{\partial t}+\frac{1}{2\rho c}\left\{\mathscr{L}_4-\alpha_{\mathscr{L}_1}-\frac{\rho\,(u-c)\,u}{c}\frac{\partial u}{\partial t}\right\}+\frac{v}{r}\frac{\partial v}{\partial \theta}+\frac{v^2}{r}+g\cos\theta=0. \tag{3.40}$$

Equation 3.40 can be re-arranged to form

$$\frac{\partial u}{\partial t}+\frac{\frac{1}{2\rho c}\left\{\mathscr{L}_4-\alpha_{\mathscr{L}_1}\right\}+\frac{v}{r}\frac{\partial v}{\partial \theta}+\frac{v^2}{r}+g\cos\theta}{1-\frac{(u-c)u}{2c^2}}=0 \tag{3.41}$$

Equation 3.41 is solved to find $\partial u/\partial t$. Equation 3.38 is then used to calculate $\mathscr{L}_1$, which is passed to equations 3.30 and 3.31, and used to calculate $\partial\rho/\partial t$ and $\partial p/\partial t$.

Prescription of $\mathscr{L}_2$

When $u > 0$ on Γ, $\mathscr{L}_2$ may be determined from the definition in equation 3.29b. When $u \leq 0$ on Γ, $\mathscr{L}_2$ must be determined based on the solution of the NLAA. Equations 3.35 and 3.36 are substituted into equation 3.29b to obtain

$$\mathscr{L}_2 = \frac{\rho u^2}{r}\left[\frac{p-p_\infty}{c\rho_\infty}-2u+\frac{u^2}{2c}\right]+\rho u\left(1-\frac{u}{c}\right)\frac{\partial u}{\partial t}. \tag{3.42}$$

Once equation 3.41 has been solved, $\mathscr{L}_2$ is determined from equation 3.42, which is used in equation 3.30 to obtain $\partial\rho/\partial t$.

$\mathscr{L}_2$ describes the entropy at the boundary. An alternative approach is to state that the entropy is constant in the radial direction by setting $\mathscr{L}_2 = 0$. I find that the maximum relative error caused by this second approach is of the order of 0.001% in the tests conducted in Chapter 5.

Prescription of $\mathscr{L}_3$

The NLAA is based on the polar and azimuthal components of velocity being zero on Γ. In a two-dimensional scheme, this may not be the case. However, as the NLAA makes no provision for determining the variation of polar velocities with radius, I make the simplest approximation, and state that the variation of polar velocity with radius is zero. Hence, if $u > 0$ on Γ, then $\mathscr{L}_3$ is determined from equation 3.29a, otherwise $\mathscr{L}_3 = 0$. This assumption is equivalent to stating that there is no advection of transverse velocities through Γ.

Prescription of $\mathscr{L}_4$

Since the motion on Γ is always subsonic, $\mathscr{L}_4$ is defined by equation 3.29d.

3.5 The validity of the NLAA boundary condition

Consider the spherical domain Ω, of finite radius, centred on the origin of a polar coordinate system and bounded by Γ. In the derivation of the NLAA, no constraints are placed on the solution within Ω. However, it is assumed that on Γ, and in the region outside Ω

1. all motion is of low Mach number: $\mathrm{Ma} < 0.1$;
2. the fluid is inviscid: $\mu = 0$;
3. all fluid properties are subject to spherical symmetry: $\partial/\partial\theta = 0$;
4. the velocity is in the radial direction only: $\mathbf{u} = u\mathbf{e}_r$.

The artificial boundary condition using the NLAA is inherently non-exact, as it is based on the asymptotic solution to the motion as advective terms approach zero. The

approximate nature of the boundary condition means that it will always be applied in regions where some of the assumptions listed above are violated. The performance of the boundary condition is determined by the magnitude of the errors introduced when these assumptions are made.

The NLAA boundary condition assumes the density of the water to be constant outside the computational domain. This assumption is justified by the low Mach number of the motion in this region. However, under certain circumstances, for instance when air guns are deployed in arrays, the arrival of reflections of multiple guns in the sea surface can cause a sudden decrease in pressure to the extent that cavitation occurs (Landrø *et al.*, 2011), referred to as ''ghost cavitation''. This cavitation causes high frequencies (above 1kHz). Further to the noise caused, the ghost cavitation constitutes a region of lower density, which would affect the propagation characteristics of the signals emitted by the air gun. This is not an issue for the present model, with a single air gun and no sea surface reflection. However, were the sea surface and more guns included in the model ghost cavitation may need to be accounted for. The present model is not capable of accounting for ghost cavitation if it occurs outside the computational domain. A solution to this would be to estimate (using the principle of superposition) where cavitation might occur and then expand the domain to include this region. The present model does not account ghost cavitation within the computational domain, although it could easily be modified to do so. So called ''single fluid'' cavitation models, such as those of Yuan *et al.* (2001), Martynov *et al.* (2006) and Goncalvés and Patella (2011) assume a locally homogeneous mixture of liquid and vapour phases, and adjust the equation of state accordingly. The finite volume scheme described in Chapter 4 could be modified to account for cavitation in this manner.

The surface of the bubble is known to be very uneven, and after the first bubble oscillation, is very much like a foam. In reality, there is a transition region between the bubble and the (pure) water containing many smaller bubbles. In this region, the density is not constant. However, although the bubble consists of many bubbles after the first oscillation, they remain connected, and continue to expand as a single bubble, as can be seen in Figure 1.2. Furthermore, any small bubbles which do become

separated constitute only a thin region near the bubble, and will be of little importance far from the bubble. As an air gun bubble rises it leaves behind a trail of smaller bubbles, and the average density in this trail is lower than in pure water. The effect of this reduced density due to bubbles on the propagation of pressure waves will be dependent on the frequencies of the pressure waves. For low frequency waves, which dominate the signals emitted by air-gun bubbles, the effect will be very slight, and in the form of a slight damping. For any high frequencies (with wavelengths a close to the characteristic length scale of theese small bubbles) present in the signal the effect will be a scattering of the signal. The reduction in density is unlikely to be significant in any case. As demonstrated by the clarity of frame (C) in Figure 1.2, the volumetric ratio of air to water around the bubble is low.

Whilst there may be small regions of lower density around the bubble, these are not present far from the bubble in the present case, and so the assumptions on which the boundary condition is based are not invalidated. The effect of reduced density in the water near the bubble is expected to be small, and a more detailed two- or three-dimensional simulation based on the present framework could be used to verify this.

The non-linear acoustic approximation was designed to be used in the form of equation 3.16 at the interface between an air gun bubble and the water. As can clearly be seen in the photographs of Langhammer and Landrø (1996), the surface of an air gun bubble is not perfectly spherical or smooth. Assumptions 3 and 4 are clearly violated at the bubble wall. However, since the wavelengths of the acoustic radiation are much larger than the diameter of the bubble, beyond a few bubble radii from the bubble, all the wrinkles shown in Langhammer's 1996 photographs, and visible at optical wavelengths, are inaudible. My approach is to apply the NLAA a small distance outside the bubble. The benefits of this approach are two-fold. Firstly, the NLAA is applied at a location where the assumptions on which it is based are more likely to be valid, as this location is further from potential sources of turbulence and asymmetric motion. Secondly, by using the NLAA to apply a boundary condition to a finite volume scheme, a non-spherical bubble can be modelled, and more complex phenomena may

be simulated. Whilst using the present scheme may be an improvement on previous air gun modelling, it is still important to quantify the errors introduced by the boundary condition. I do this in Chapters 5 and 6.

In Chapter 5 I demonstrate that the boundary condition does not introduce significant errors due to assumption 1 in cases of interest, and show the limitations of the method due to the violation of this assumption. There are no viscous terms in the Euler equations, so assumption 2 remains valid. If viscous terms are included and instead the Navier-Stokes equations are solved, spurious reflections in vorticity might occur due to the boundary conditions if the solution on Γ were to contain sufficient vorticity. In this case, it is important to position Γ adequately far from potential sources of vorticity, such as the surface of a bubble, or solid surfaces. In Chapter 7 I show the effects of including viscous terms. However, it should be born in mind that finite volume schemes require a certain amount of numerical viscosity in order to remain stable, and the solution within Ω can never be completely inviscid. This may have an effect on the boundary condition; however, I do not attempt to disentangle errors due to this effect from errors discussed in Chapter 5.

For one-dimensional simulations, assumptions 3 and 4 are constrained to be valid. In the two-dimensional case, they are not constrained to be valid. Indeed, the purpose of an extension to two dimensions is to allow the simulation of asymmetric features of bubbles, such as the translation and deformation due to the effects of gravity. It is important that the boundary condition is applied only at locations far enough from the region of interest that these assumptions do not introduce significant errors. The results in Chapter 6 demonstrate that limits on the performance of the NLAA boundary condition are, in cases of intereste, likely not to be imposed by the violation of these assumptions, but by the constraints of assumption 1, and the limitations of the numerical scheme described in the next chapter.

3.6 Summary

In this chapter I derive a new artificial boundary condition for numerical simulations of oscillating bubbles and similar problems on a finite domain. The method is applicable when the problem is spherical in nature and close to spherically symmetric, and the motion at the domain boundary is of low Mach number (less than 0.1). The boundary condition is based on the non-linear acoustic approximation (NLAA), developed for use in modelling seismic air guns. In this chapter I present a brief derivation of the NLAA, and the well known characteristic boundary condition formalism. I use the NLAA to calculate an approximate solution to the motion outside the domain based on the solution at the domain boundary. I apply boundary conditions by using the approximate solution to describe all characteristic waves incoming to the domain at the boundary. I discuss the validity of the NLAA boundary condition, which is tested in subsequent chapters.

Chapter 4

Computational implementation

4.1 Introduction

The boundary condition described in Chapter 3 is designed to be applied to finite volume simulations of the Euler equations. The Euler equations are a system of non-linear hyperbolic partial differential equations, which describe the behaviour of compressible inviscid fluids. One of the most commonly used class of approaches for the numerical solution of the Euler equations is finite volume methods. In Chapter 2 I introduce finite volume methods. To recapitulate from Chapter 2, finite volume methods are generally based around the following framework. The spatial domain is discretised into a set of discrete cells, and fluid properties (such as density or momentum) at time t are approximated by a set of discrete values in each cell in the discretised domain. A reconstruction procedure is used to calculate approximations to the fluxes of properties between cells. These fluxes are used, with the governing equations, to obtain the time derivatives of fluid properties, which are then numerically integrated to obtain the solution at time $t + \delta t$. This process completes one time-step. Reviews and discussion of early finite difference and finite volume schemes are given by, for example, Sod (1978), Colella and Woodward (1984), Colella (1985), Einfeldt (1988) and the references therein.

In this chapter I describe the numerical scheme I use to test the NLAA boundary condition, and to obtain all results in the subsequent chapters. Throughout this chapter, I present the scheme in two dimensions. One-dimensional simulations are achieved using the same scheme, but with the constraint that variation and motion are in the radial direction only. In Section 4.2 I present the Euler equations in two dimensions, and define the computational domain. In Section 4.3 I provide details of the single phase finite volume scheme. In Section 4.4 I describe the method for setting the time step. In Sections 4.5 and 4.6 I describe the method used to simulate the interface, and discuss the choice of interface modelling scheme, with brief results comparing the performance of different methods. In section 4.7 I describe the application of the boundary condition. In Section 4.9 I discuss the performance and limitations of the scheme.

4.2 Governing equations and computational domain

In polar coordinates with polar axi-symmetry, the two-dimensional Euler equations may be written

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}\left(\mathbf{U}\right)}{\partial r} + \frac{\partial \mathbf{G}\left(\mathbf{U}\right)}{\partial \theta} + \mathbf{S_r}\left(\mathbf{U}\right) + \mathbf{S}_\theta\left(\mathbf{U}\right) = \mathbf{D}\left(\mathbf{U}\right), \tag{4.1}$$

where $\mathbf{U}$ is a vector of conservative variables, and $\mathbf{F}$ and $\mathbf{G}$ are vectors of fluxes, given by

$$\mathbf{U} = \left[\begin{array}{cccc} \rho, & \rho u, & \rho v, & E \end{array} \right]^T, \tag{4.2}$$

$$\mathbf{F} = \left[\begin{array}{cccc} \rho u, & \rho u^2 + p, & \rho uv, & u\left(E + p\right) \end{array} \right]^T \tag{4.3}$$

and

$$\mathbf{G} = \left[\begin{array}{cccc} \rho v, & \rho uv, & \rho v^2 + p, & v\left(E + p\right) \end{array} \right]^T, \tag{4.4}$$

in which ρ, u, v, E and p are the density, radial velocity, polar velocity, total energy and pressure respectively. The source terms $\mathbf{S_r}$ and $\mathbf{S}_\theta$ are due to the divergence of the polar coordinate system, and are given by

$$\mathbf{S_r} = \frac{2}{r}\left[\begin{array}{cccc} \rho u, & \rho u^2, & \rho uv, & u\left(E + p\right) \end{array} \right]^T \tag{4.5}$$

and

$$\mathbf{S}_\theta = \frac{1}{r\tan\theta}\left[\begin{array}{cccc} \rho v, & \rho uv, & \rho v^2, & v\left(E + p\right) \end{array} \right]^T. \tag{4.6}$$

The effects of gravity are accounted for by $\mathbf{D}$, defined by

$$\mathbf{D} = \left[\begin{array}{cccc} 0, & -\rho g \cos\theta, & \rho g \sin\theta, & -\rho g \left(u\cos\theta - v\sin\theta \right) \end{array} \right]^T, \tag{4.7}$$

where $g = 9.81\text{ms}^{-2}$ is gravity, such that the gravitational forces are in a direction aligned with the polar axis where $\theta = \pi$ (or $-z$ in a Cartesian frame of reference as described in the section on notation at the start of this thesis). The equations are closed with a stiffened gas equation of state given by

$$p = \left(\gamma - 1\right)\rho e - \gamma p_c, \tag{4.8}$$

where the specific energy e is related to the total energy by $E = \rho e + \frac{1}{2}\rho u^2$. For air $\gamma = 1.4$ and $p_c = 0$, which is equivalent to the ideal gas equation of state. For water, typically $\gamma = 7.0$ and $p_c = 3 \times 10^8$ Pa.

I numerically solve equations 4.1 on a finite spherical domain Ω, with radius R_D, centred on the origin of a polar coordinate system. I denote by Γ the boundary of the domain, on which $r = R_D$. The domain is subject to symmetry about the polar axis, and hence the region for computation is defined by the set of points $\mathbf{r} \in \left([0, R_D], [0, \pi]\right)$. The domain is discretised into $m \times n$ cells, with m cells with side length δr in the radial direction and n cells with side length $r\delta\theta$ in the polar direction. Two-phase flow simulations are achieved using a single phase finite volume Euler solver in combination with a ghost fluid method to account for the air-water interface.

4.3 Single phase Euler solver

The single phase Euler solver is a dimensionally-split first order Godunov-type scheme (Godunov, 1959) (The original work by Godunov is in Russian. Useful expanations of Godunov-type schemes are given by, for example, Einfeldt (1988) and Ivings *et al.* (1998).). Spatial reconstruction is piecewise constant in each cell, based on cell centre values $\mathbf{U}^n_{i,j}$, where i and j denote the spatial indices of the cell in the radial and polar directions respectively, and n denotes the time index. The time index at $t = 0$ is $n = 1$. I describe the scheme for an arbitrary time step n. The solution $\mathbf{U}^n$ at the n-th time step

is known. Riemann problems at the cell edges are defined by $R_{i+\frac{1}{2},j} = R\left(\mathbf{U}^n_{i,j}, \mathbf{U}^n_{i+1,j}\right)$, and $R_{i,j+\frac{1}{2}} = R\left(\mathbf{U}^n_{i,j}, \mathbf{U}^n_{i,j+1}\right)$, and are solved using a Roe-average Riemann solver due to Hu *et al.* (2009) to obtain HLLC fluxes $\hat{\mathbf{F}}_{i+\frac{1}{2},j}$ and $\hat{\mathbf{G}}_{i,j+\frac{1}{2}}$ (Toro *et al.* (1994)). Details of the approximate Riemann solver and HLLC fluxes are given in Appendix A. Time integration is first-order. The scheme is dimensionally split, and the geometric and gravitational source terms are applied using a basic operator splitting technique. Hence, a single time step involves solving the following sequence of equations:

$$\mathbf{U}^{(1)}_{i,j} = \mathbf{U}^n_{i,j} - \frac{\delta t}{\delta r}\left[\hat{\mathbf{F}}_{i+\frac{1}{2},j}\left(\mathbf{U}^n\right) - \hat{\mathbf{F}}_{i-\frac{1}{2},j}\left(\mathbf{U}^n\right)\right], \tag{4.9a}$$

$$\mathbf{U}^{(2)}_{i,j} = \mathbf{U}^{(1)}_{i,j} - \delta t \mathbf{S_r}\left(\mathbf{U}^{(1)}_{i,j}\right), \tag{4.9b}$$

$$\mathbf{U}^{(3)}_{i,j} = \mathbf{U}^{(2)}_{i,j} - \frac{\delta t}{\delta \theta}\left[\hat{\mathbf{G}}_{i,j+\frac{1}{2}}\left(\mathbf{U}^{(2)}\right) - \hat{\mathbf{G}}_{i,j-\frac{1}{2}}\left(\mathbf{U}^{(2)}\right)\right], \tag{4.9c}$$

$$\mathbf{U}^{(4)}_{i,j} = \mathbf{U}^{(3)}_{i,j} - \delta t \mathbf{S}_\theta\left(\mathbf{U}^{(3)}_{i,j}\right), \tag{4.9d}$$

$$\mathbf{U}^{n+1}_{i,j} = \mathbf{U}^{(4)}_{i,j} + \delta t \mathbf{D}\left(\mathbf{U}^{(4)}_{i,j}\right), \tag{4.9e}$$

where a superscript in parentheses denotes the solution to an intermediate step. Starting from $\mathbf{U}^n$, the fluxes $\hat{\mathbf{F}}$ in the radial direction, are calculated, and then used to obtain the intermediate solution $\mathbf{U}^{(1)}$ from equation 4.9a. $\mathbf{U}^{(1)}$ is then used to calculate the source terms $\mathbf{S_r}$, and then equation 4.9b is used to obtain $\mathbf{U}^{(2)}$. $\mathbf{U}^{(2)}$ is then used to determine the fluxes $\hat{\mathbf{G}}$ in the polar direction, which are used with equation 4.9c to calculate $\mathbf{U}^{(3)}$. $\mathbf{U}^{(3)}$ is used to determine the source terms $\mathbf{S}_\theta$, which are then used in equation 4.9d to obtain $\mathbf{U}^{(4)}$. $\mathbf{U}^{(4)}$ is then used to determine the graviational source terms, $\mathbf{D}$. Finally equation 4.9e is used to obtain $\mathbf{U}^{n+1}$. This first-order operator splitting technique is adequate in the present implementation, as the underlying Godunov-type scheme is only first-order accurate in space in and time. Higher order techniques, such as the second order Strang-splitting (Strang, 1968), show no significant improvement in results, whilst increasing computational costs.

4.4 Courant condition

The time step is determined by the Courant condition (Courant *et al.* (1928), or Courant *et al.* (1967) for an English translation), also known as the Courant-Friedrichs-Lewy or CFL condition:

$$\delta t = \mathrm{CFL}\left[\max\left(\frac{u_{sig}}{\delta r} + \frac{v_{sig}}{r\delta\theta}\right)\right]^{-1}, \tag{4.10}$$

where CFL is known as the CFL number, $u_{sig} = c + |u|$ and $v_{sig} = c + |v|$. The speed of sound is calculated according to $c^2 = \gamma\left(p + p_c\right)/\rho$. CFL > 0, and for stability CFL ≤ 1. If CFL $= 1$, equation 4.10 sets the time step as the minimum time interval required for the fastest characteristic to cross a computational cell. In all simulations in this thesis CFL $= 0.8$.

4.5 Level-set

Level set methods are a class of methods for tracking moving interfaces, first introduced by Osher and Sethian (1988). A variable ϕ is introduced as a signed distance function of an interface, such that the region $\phi \leq 0$ contains exclusively one fluid, and the region $\phi > 0$ contains exclusively another fluid. The level set is advected with the local fluid velocity, and the zero level set (where $\phi = 0$) remains on the interface. The level set ϕ is advected using the level set equation

$$\frac{\partial \phi}{\partial t} + \mathbf{u}_{LS} \cdot \nabla \phi = 0. \tag{4.11}$$

Spatial derivatives of the level set are obtained using an upwind discretisation based on a WENO reconstruction procedure due to Borges *et al.* (2008), details of which are given in Appendix A. For one-dimensional simulations, every point on the interface has the same velocity, and I set $\mathbf{u}_{LS}$ everywhere equal to the velocity at the interface, thus maintaining ϕ as a signed distance function of the interface. For two-dimensional simulations, the velocity of the interface varies along the interface, and I set $\mathbf{u}_{LS}$ equal to the extrapolated interface velocity obtained for the ghost fluid method (details of which I give in the next section).

Information about the gradient of the level set is required near the interface. In the presence of velocity gradients equation 4.11 alone fails to maintain the level set as a signed distance function, and so the level set is re-initialised after each update according to the equation

$$\frac{\partial \phi}{\partial \tau} = \operatorname{sgn}\left(\phi^0\right)\left(1 - |\nabla \phi|\right), \tag{4.12}$$

in which ϕ^0 is the solution to equation 4.11, and τ represents an artificial time. Equation 4.12 can be expressed as a scalar convection equation

$$\frac{\partial \phi}{\partial \tau} + \operatorname{sgn}\left(\phi^0\right)\frac{\nabla \phi}{|\nabla \phi|} \cdot \nabla \phi = \operatorname{sgn}\left(\phi^0\right). \tag{4.13}$$

In this form it can be seen that deviations from a signed distance function of ϕ^0 are advected away from the interface with velocity $\operatorname{sgn}\left(\phi^0\right)\frac{\nabla \phi}{|\nabla \phi|}$. If ϕ is a signed distance function then $|\nabla \phi| = 1$ and $\frac{\partial \phi}{\partial \tau} = 0$. If equation 4.12 is solved until convergence, the solution is a signed distance function of the interface, as defined by the zero of ϕ^0. As information about $\nabla \phi$ is only needed in the vicinity of the interface, I do not solve equation 4.12 to convergence. I find that solving equation 4.12 for $0.4 \max(m, n)$ artificial time steps is sufficient. The solution of equation 4.12 is computationally expensive. In order to preserve the location of the zero level set, and to avoid the transmission of information across the interface, equation 4.12 must be solved using an upwind discretisation (upwind in the sense of the advection velocity $\operatorname{sgn}\left(\phi^0\right)\frac{\nabla \phi}{|\nabla \phi|}$), which is problematic close to the interface. Various authors have worked the development of efficient and accurate algorithms for level set re-initialisation (for example Adalsteinsson and Sethian (1995), Russo and Smereka (2000), Spelt (2005) and references therein). I use a scheme based on that due to Russo and Smereka (2000), which is first-order accurate over the whole domain. Note that the re-initialisation procedure is not necessary in one-dimensional simulations due to the choice of $\mathbf{u}_{LS}$. In the two-dimensional cases, the level set velocity is set equal to the extrapolated interfacial velocity obtained using the ghost fluid method. For cells which bound the interface, this velocity is equal to the interface velocity. Hence, cells which border the interface do not need to be re-initialised.

The level set re-initialisation procedure I use for cells which do not border the interface

is as follows. A mollified sign function of ϕ^0, $S\left(\phi^0\right)$, is calculated as

$$S\left(\phi^0\right) = \frac{\phi^0}{\left(\left(\phi^0\right)^2 + \varepsilon^2\right)^2}, \tag{4.14}$$

where ε is a small number, typically a characteristic length scale of the computational mesh. I use $\varepsilon = \delta r$. I define

$$a = \left(\phi_{i,j}^{(n)} - \phi_{i-1,j}^{(n)}\right)/\delta r, \tag{4.15a}$$

$$b = \left(\phi_{i+1,j}^{(n)} - \phi_{i,j}^{(n)}\right)/\delta r, \tag{4.15b}$$

$$c = \left(\phi_{i,j}^{(n)} - \phi_{i,j-1}^{(n)}\right)/r\delta\theta, \tag{4.15c}$$

$$d = \left(\phi_{i,j+1}^{(n)} - \phi_{i,j}^{(n)}\right)/r\delta\theta, \tag{4.15d}$$

where the superscript (n) denotes the index of the artificial time during the reconstruction procedure, and should not be confused with the time index of the overall scheme. Let $f_+ = \max(f, 0)$ and $f_- = \min(f, 0)$. Then

$$G\left(\phi\right)_{i,j} = \begin{cases} \sqrt{\max\left(a_+^2, b_-^2\right) + \max\left(c_+^2, d_-^2\right)} - 1 & \text{if } \phi_{i,j}^0 > 0, \\ \sqrt{\max\left(a_-^2, b_+^2\right) + \max\left(c_-^2, d_+^2\right)} - 1 & \text{if } \phi_{i,j}^0 \leq 0. \end{cases} \tag{4.16}$$

Finally

$$\phi_{i,j}^{(n+1)} = \begin{cases} \phi_{i,j}^{(n)} & \text{if } (i,j) \in \Sigma_{int}, \\ \phi_{i,j}^{(n)} - \delta\tau S\left(\phi_{i,j}^0\right) G\left(\phi\right)_{i,j} & \text{otherwise,} \end{cases} \tag{4.17}$$

where Σ_{int} is the set of cells which bound the interface.

4.6 Ghost fluid method

I model the interface using a ghost fluid method (GFM). In this section I summarise GFMs, discuss the appropriate choice of ghost fluid method for the present work, and provide details of the application of the ghost fluid method in two dimensions. Ghost fluid methods are a family of front-tracking methods for the simulation of multimedia flows with sharp interfaces, first developed by Fedkiw *et al.* (1999a) as a natural extension of earlier work on the interaction between moving solid surfaces and fluids.

The location of the material interface is tracked, usually with a level set, and sets of ghost cells are used around the interface to prescribe the correct boundary conditions for each fluid.

At each time step the location of the interface is determined by finding the location of the zero level set. The two-fluid domain Ω, is duplicated to create two one-fluid domains, Ω_1 and Ω_2. Ω_1 contains real cells where $\phi \leq 0$ and ghost cells where $\phi > 0$. Ω_2 contains ghost cells where $\phi \leq 0$ and real cells where $\phi > 0$. The properties in the real cells are copied directly from the properties in Ω. There are various methods of specifying the properties in ghost cells, which fall into two general classes: (1) methods which extrapolate discontinuous properties across the interface into the ghost cells and (2), methods which use Riemann problems at the interface to define the properties in the ghost cells. Details of ghost cell specification for various versions of the ghost fluid method are given below. Once the ghost cells are defined, the properties in Ω_1 and Ω_2 are updated separately for a single time step using the single phase Euler solver, whilst the level set is updated to obtain the new interface position. The properties in Ω are then reconstructed from the two one-fluid domains, based on the sign of the level set.

Different versions of ghost fluid method

The choice of GFM has a significant impact on the results of the simulation, as it is the GFM which governs the transmission of momentum and energy through the interface. The first ghost fluid method of Fedkiw *et al.* (1999a) simply copies the values of the continuous variables (pressure and velocity normal to the interface) onto ghost cells directly from the real cells, and extrapolates the discontinuous variables (density or entropy, and velocity components tangential to the interface) across the interface. The transmission of shock waves through material interfaces is highly dependant on the material properties either side of the interface, and some cases, such as those with very different materials either side of the interface, the basic extrapolation and copying procedure of Fedkiw *et al.* (1999a) is insufficient. For this reason the GFM of Fedkiw *et al.* (1999a) does not provide accurate results for air-water interfaces, and I omit this

version from the following discussion as it is not applicable to the air-water interfaces considered in the present work. Fedkiw *et al.* (1999a,b) also develop the 'isobaric fix' for the GFM, in which the last real cell on either side of the interface is also treated as a ghost cell, a trick designed to avoid the problem of overheating near the interface. Various other versions of GFM have since been developed. Liu *et al.* (2003) propose a GFM, capable of simulating gas-water interfaces, and in Liu *et al.* (2005), they provide an analysis of ghost fluid methods for gas-water simulations. Fedkiw (2002) also presents a modified ghost fluid method (mGFM), which accounts by construction for the fact that at an air-water interface it is the air pressures and water velocites which dominate the motion. The 'real ghost fluid method' (rGFM) was developed by Wang *et al.* (2006), in which a two-phase Riemann problem is constructed at the interface, and the solution of this Riemann problem is used to define the ghost states. The method of Wang *et al.* (2006) is similar to the 'interface interaction' method of Hu and Khoo (2004). More recently, methods such as the conservative interface methods in Hu *et al.* (2006, 2009) have been developed, in which the transmission of momentum and energy across the interface is accounted for using source terms. For the one-dimensional results in Chapter 5 I use a version of the rGFM of Wang *et al.* (2006); however, I omit the isobaric fix, and do not modify real cells. Below I briefly discuss the merits and drawbacks of various versions of the GFM. For clarity, I describe how each version is constructed in one dimension only. I give details of the chosen version in two dimensions later.

Let the fluid properties in cell i at the start of a time step be denoted $\mathbf{U}_i^n$. Let the cell immediately to the left of the interface (containing fluid 1, which in the present case is air) have index $i = q$, and hence the cell immediately to the right of the interface (containing fluid 2, which in the present case is water) is cell $q + 1$. q is found by searching for the q which satisfies $\phi_q \phi_{q+1} \leq 0$. The properties in the two single-fluid domains are determined differently for the different versions of GFM, as described below, and are denoted $\mathbf{U}_{\Omega_1,i}^n$ and $\mathbf{U}_{\Omega_2,i}^n$. Ω_1 contains real cells where $\phi \leq 0$ ($i \leq q$), and a band of ghost cells where $\phi > 0$ ($i \geq q + 1$). Ω_2 is populated by real cells where $\phi > 0$ and a band of ghost cells where $\phi \leq 0$. The band of ghost cells is required to

be a minimum thickness of 2 cells for first-order methods, with higher-order methods requiring more ghost cells. The single-phase Euler solver is used to update each of the domains separately, yielding $\mathbf{U}_{i,\Omega_1}^{n+1}$ and $\mathbf{U}_{i,\Omega_2}^{n+1}$. The level set is updated, then the two one-fluid domains are recombined to give the updated properties in the two-fluid domain according to

$$\mathbf{U}_i^{n+1} = \begin{cases} \mathbf{U}_{i,\Omega_1}^{n+1} & \text{if } \phi_i^{n+1} \leq 0 \\ \mathbf{U}_{i,\Omega_2}^{n+1} & \text{if } \phi_i^{n+1} > 0. \end{cases} \tag{4.18}$$

Real ghost fluid method (rGFM)

In the rGFM of Wang *et al.* (2006), the ghost cells are prescribed by

$$\mathbf{U}_{\Omega_1,i}^n = \begin{cases} \mathbf{U}_i^n & \text{if } i \leq q-1, \\ \mathbf{U}_L^\star & \text{if } i \geq q, \end{cases} \tag{4.19a}$$

$$\mathbf{U}_{\Omega_2,i}^n = \begin{cases} \mathbf{U}_R^\star & \text{if } i \leq q+1, \\ \mathbf{U}_i^n & \text{if } i \geq q+2, \end{cases} \tag{4.19b}$$

where $\mathbf{U}_L^\star$ and $\mathbf{U}_R^\star$ are the solution to the Riemann problem defined by $R\left(\mathbf{U}_{q-1}^n, \mathbf{U}_{q+2}^n\right)$. The Riemann problem may be solved using any approximate or exact Riemann solver. I use a form of that due to Hu *et al.* (2009), described in Appendix A.

It is worth considering the choices of Riemann problem input state, and the decision to treat the last real cells as ghost cells. With this method the interface effectively has a width of three computational cells. The solution in the second cell from the interface in each fluid has a dependence on the solution at the previous time step in the second cell from the interface in the other fluid. Information is able to travel from $q-1$ to $q+2$ and vice versa in a single time step, which violates the CFL condition. This argument applies to any of the versions of ghost fluid method which include the isobaric fix.

Basic real ghost fluid method (basic rGFM)

This version of the rGFM is the one I use for one-dimensional simulations, including all results in Chapter 5, which is perhaps the most simple form of Riemann problem-based ghost fluid method. This GFM appears to be equivalent to the interface interaction method of Hu and Khoo (2004), although expressed in a different form. This version was chosen after consideration of the arguments against the rGFM of Wang *et al.* (2006) put forward above.

Ghost cell states are prescribed by

$$\mathbf{U}^n_{\Omega_1,i} = \begin{cases} \mathbf{U}^n_i & \text{if } i \leq q, \\ \mathbf{U}^\star_L & \text{if } i \geq q+1, \end{cases} \tag{4.20a}$$

$$\mathbf{U}^n_{\Omega_2,i} = \begin{cases} \mathbf{U}^\star_R & \text{if } i \leq q, \\ \mathbf{U}^n_i & \text{if } i \geq q+1, \end{cases} \tag{4.20b}$$

where $\mathbf{U}^\star_L$ and $\mathbf{U}^\star_R$ are the solution to the Riemann problem defined by $R\left(\mathbf{U}^n_q, \mathbf{U}^n_{q+1}\right)$. For the one-dimensional scheme, I set $\mathbf{u}_{LS} = (u^\star, 0)^T$ here.

Modified ghost fluid method (mGFM)

The mGFM due to Fedkiw (2002) is described by

$$\mathbf{U}^n_{\Omega_1,i} = \begin{cases} \mathbf{U}^n_i & \text{if } i \leq q, \\ (\rho_q,\, p_q,\, u_i) & \text{if } i \geq q+1, \end{cases} \tag{4.21a}$$

$$\mathbf{U}^n_{\Omega_2,i} = \begin{cases} (\rho_{q+1},\, p_i,\, u_{q+1}) & \text{if } i \leq q, \\ \mathbf{U}^n_i & \text{if } i \geq q+1. \end{cases} \tag{4.21b}$$

Modified ghost fluid method with isobaric fix

The mGFM with an isobaric fix is described by

$$\mathbf{U}^n_{\Omega_1,i} = \begin{cases} \mathbf{U}^n_i & \text{if } i \leq q-1, \\ (\rho_{q-1},\, p_{q-1},\, u_i) & \text{if } i \geq q, \end{cases} \tag{4.22a}$$

$$\mathbf{U}^n_{\Omega_2,i} = \begin{cases} (\rho_{q+2},\, p_i,\, u_{q+2}) & \text{if } i \leq q+1, \\ \mathbf{U}^n_i & \text{if } i \geq q+2. \end{cases} \tag{4.22b}$$

Note that the same arguments regarding the speed of information transmission across the interface apply to this version of the GFM as for the rGFM.

Modified real ghost fluid method (mrGFM)

For two-dimensional simulations I use a version of the rGFM, which I refer to as the modified real ghost fluid method (mrGFM), described by

$$\mathbf{U}^n_{\Omega_1,i} = \begin{cases} \mathbf{U}^n_i & \text{if } i \leq q, \\ (\rho^\star_L,\, p_q,\, u^\star) & \text{if } i \geq q+1, \end{cases} \tag{4.23a}$$

$$\mathbf{U}^n_{\Omega_2,i} = \begin{cases} (\rho^\star_R,\, p_q,\, u^\star) & \text{if } i \leq q, \\ \mathbf{U}^n_i & \text{if } i \geq q+1, \end{cases} \tag{4.23b}$$

where the star-properties are from the solution to the Riemann problem defined by $R\left(\mathbf{U}^n_q, \mathbf{U}^n_{q+1}\right)$. This version is a combination of the basic rGFM of Wang *et al.* (2006) and the mGFM of Fedkiw (2002). It provides increased stability over the rGFM and basic rGFM in two-dimensions, and avoids an explicit dependence of ghost cell states on water pressures, which are known to contain errors near the interface when using polar coordinate systems (see Appendix B for details).

A comparison of various ghost fluid methods

In order to compare the different forms of ghost fluid method, I present the results to test problems in one-dimension. These test problems are the classic Sod shock tube problem (Sod, 1978), and an air-water shock tube problem related to the underwater explosion problem of Flores and Holt (1981). These test problems are performed on a grid with resolution $\delta r = 0.005$, unless otherwise stated, and with a CFL number of 0.8. Note that these problems are solved in a one-dimensional Cartesian framework, by setting all geometric source terms to zero, as exact solutions to both problems are available in Cartesian coordinates.

Sod shock tube

The Sod shock tube problem is very often the first test considered when developing a numerical scheme for hydrodynamic simulations. The problem was first introduced by Sod (1978), and is a theoretical initial value problem for the Euler equations, consisting of two ideal gases at rest, separated by a discontinuity. More details of this problem are given in Chapter 2. The initial conditions are

$$(\rho,\, u,\, p,\, \gamma) = \begin{cases} (1,\, 0,\, 1,\, 1.4) & \text{if } r \leq 0.5, \\ (0.125,\, 0,\, 0.1,\, 5/3) & \text{if } r > 0.5. \end{cases} \tag{4.24}$$

At $t = 0$ the two gases are allowed to interact. The resulting solution contains a rarefaction wave travelling left and a contact discontinuity and a shock wave travelling right. The exact solution can be obtained by using an exact Riemann solver. I use a Riemann solver due to Ivings *et al.* (1998), and calculate the solution at a resolution of $\delta r = 10^{-3}$. Figure 4.1 shows the density and velocity profiles obtained using the numerical scheme in one dimension for various versions of the ghost fluid method. Note that the rarefaction and shock waves are not as sharp in the numerical results as they are in the exact solution, due to the relatively coarse grid and the use of only a first-order finite volume scheme. The shock wave is of approximately the correct amplitude for all versions of GFM. For the mGFM both with and without the isobaric fix, the

shock speed is high compared with the exact solution. All versions of GFM are subject to some overshoot around the contact discontinuity; however, the overshoot is much greater for the mGFM than the various rGFMs. Also note the spurious overshoot in the rarefaction wave for the mGFM with isobaric fix. It is clear from Figure 4.1 that the Riemann problem-based GFMs outperform the mGFM, even in this basic test case.

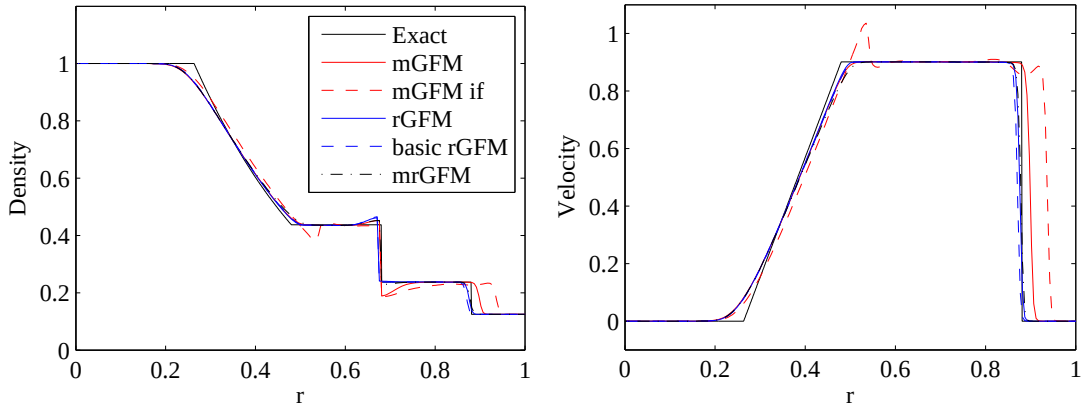


Figure 4.1: Spatial density and velocity profiles for the Sod shock tube problem at time $t = 0.2$ for various versions of the ghost fluid method.

Air-water shock tube

The next problem I use to test the ghost fluid method is an air-water shock tube problem based on the underwater explosion problem of Flores and Holt (1981). The initial conditions are

$$(\rho,\, u,\, p,\, \gamma,\, p_c) = \begin{cases} (1.63,\, 0,\, 83810,\, 1.4,\, 0) & \text{if } r \leq 0.5, \\ (1.025,\, 0,\, 10,\, 5,\, 4921.15) & \text{if } r > 0.5. \end{cases} \tag{4.25}$$

Note that these initial conditions have been dimensionally scaled, such that density has units of grammes per cubic centimeter, pressure has units of Bar (10^5Pa) and velocity has units of 10ms^{-1}. An exact solution for this problem can, in theory, be obtained using an exact Riemann solver. However, the exact Riemann solver of Ivings *et al.* (1998) fails to converge on the correct solution for this problem (in fact, it fails to converge on any solution), so I take grid-converged results obtained using the basic rGFM with $\delta r = 10^{-4}$ as a reference solution. With this level of mesh resolution,

the differences between the forms of GFM are negligible, and the smearing of pressure waves is significantly reduced.

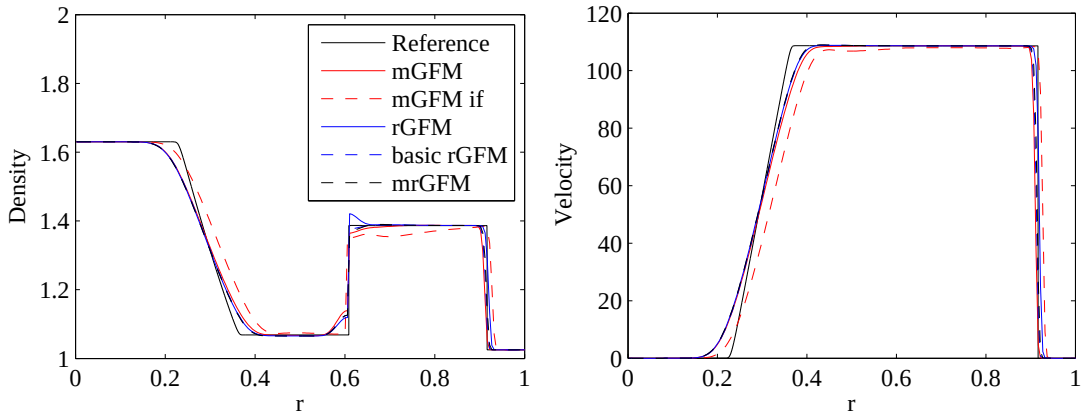


Figure 4.2: Spatial density and velocity profiles for the air-water shock tube problem at time $t = 1$ms for various versions of the ghost fluid method.

Figure 4.2 shows the density and velocity profiles of the solution to the air-water shock tube problem for various versions of ghost fluid method. As in the previous case, the mGFM with isobaric fix yields incorrect shock speeds. The shock and rarefaction wave speeds of the other versions of GFM are all in relatively good agreement with each other and with the grid converged solution. The rGFM causes overheating in the water beside the interface, whilst all other versions of GFM give slight underheating. From this test case, there are not significant differences visible in performance of the mGFM, the basic rGFM and the mrGFM.

In Chapter 5 I use the basic rGFM, as it seems the most elegant form of the GFM, and yields good results compared with the other versions of GFM tested. The exact choice of GFM is not crucial in this work, as the focus is on the efficacy of the boundary conditions. I find that for the two-dimensional model, the ghost fluid method used in Chapter 5 becomes unstable when the interface motion is not alligned with the computational mesh. Hence, I devised the modified rGFM as described above. This GFM combines the rGFM with the theory from the mGFM that air pressures and water velocites dominate the dynamics of the interface. My modified rGFM provides slightly increased dissipation in the velocity fields through the interface, leading to a

more stable simulation, whilst retaining most of the characteristics of the basic rGFM. This method removes the explicit dependence of the interface pressure on the pressure in the water, which is known to contain significant errors, as discussed in Appendix B. When running the code in two-dimensions I use the modified rGFM.

The ghost fluid method in two dimensions

In two dimensions the mrGFM involves the following steps.

1. Create two arrays of primitive variables, $\tilde{\mathbf{U}}^n_{\Omega_1}$ and $\tilde{\mathbf{U}}^n_{\Omega_2}$. Populate the parts of these arrays containing real cells according to

$$\tilde{\mathbf{U}}^n_{\Omega_1,i,j} = \begin{cases} \left(\rho^n_{i,j},\, u^n_{i,j},\, v^n_{i,j},\, p^n_{i,j}\right)^T & \text{if } \phi^n_{i,j} \leq 0 \\ (0,\, 0,\, 0,\, 0)^T & \text{otherwise,} \end{cases} \tag{4.26}$$

$$\tilde{\mathbf{U}}^n_{\Omega_2,i,j} = \begin{cases} \left(\rho^n_{i,j},\, u^n_{i,j},\, v^n_{i,j},\, p^n_{i,j}\right)^T & \text{if } \phi^n_{i,j} > 0 \\ (0,\, 0,\, 0,\, 0)^T & \text{otherwise.} \end{cases} \tag{4.27}$$

2. Determine whether each cell is beside the interface, and if so label it an 'interface cell'. A cell A with indices i_A, j_A, is an interface cell if there is a cell B such that

$$i_A - 1 \leq i_B \leq i_A + 1, \tag{4.28}$$

$$j_A - 1 \leq j_B \leq j_A + 1, \tag{4.29}$$

and

$$\phi_A \phi_B \leq 0. \tag{4.30}$$

3. For every interface cell A

   (a) find the partner cell B which satisfies equations 4.28, 4.29 and 4.30, and minimises the angle between the level set normals of the two cells, by finding B which minimises $(1 - \nabla\phi_A \cdot \nabla\phi_B)$;

(b) determine the components of the velocity in directions normal and tangent to the interface, u_n and u_t, in A and B, where the normal to the interface is positive in the direction from fluid 1 to fluid 2;

(c) solve a Riemann problem defined by $R\left(\tilde{\mathbf{U}}_A, \tilde{\mathbf{U}}_B\right)$, where $\tilde{\mathbf{U}_K} = (\rho, u_n, u_t, p)_K^T$ for $K = A, B$, obtaining star states $\mathbf{U}_L^\star$ and $\mathbf{U}_R^\star$. Find the components of the star state velocities in the radial and polar directions;

(d) use the star states of the Riemann problem to define the densities and velocities in $\tilde{\mathbf{U}}_{\Omega_1,A}^n$ if $\phi_A^n > 0$ and $\tilde{\mathbf{U}}_{\Omega_2,A}^n$ if $\phi_A^n \leq 0$. Set the pressure component of $\tilde{\mathbf{U}}_{\Omega_1,A}^n$ equal to p_B^n if $\phi_A^n > 0$. Set the pressure component of $\tilde{\mathbf{U}}_{\Omega_2,A}^n$ equal to p_A^n if $\phi_A^n \leq 0$.

4. Extrapolate the primitive properties away from the interface in the ghost regions by advecting with the level set normal:

$$\frac{\partial \xi}{\partial \tau} + \operatorname{sgn}(\phi)\, \nabla\phi \cdot \nabla\xi = 0, \tag{4.31}$$

where

$$\xi = \begin{cases} \tilde{\mathbf{U}}_{\Omega_1}^n & \text{if } \phi^n > 0 \\ \tilde{\mathbf{U}}_{\Omega_2}^n & \text{if } \phi^n \leq 0 \end{cases} \tag{4.32}$$

Equation 4.31 is solved to update *only cells which are not interface cells*, with first-order upwind discretisation of spatial derivatives and a first-order Euler method for time integration. $0.2 \max(m, n)$ artificial time steps are sufficient to obtain the required band of ghost cells.

5. Use $\tilde{\mathbf{U}}_{\Omega_1}^n$ and $\tilde{\mathbf{U}}_{\Omega_2}^n$, together with the appropriate values of γ and p_c in equation 4.8 to obtain the conservative properties in the ghost regions, $\mathbf{U}_{\Omega_1}^n$ and $\mathbf{U}_{\Omega_2}^n$. The level set velocity, $\mathbf{u}_{LS}$ is set equal to the extrapolated values, such that

$$\mathbf{u}_{LS} = \begin{cases} \left(u_{\Omega_2}^n, v_{\Omega_2}^n\right)^T & \text{if } \phi_{i,j}^n \leq 0 \\ \left(u_{\Omega_1}^n, v_{\Omega_1}^n\right)^T & \text{if } \phi_{i,j}^n > 0 \end{cases} \tag{4.33}$$

6. Update the properties in each domain separately using the single-phase Euler solver, to obtain $\mathbf{U}_{\Omega_1}^{n+1}$ and $\mathbf{U}_{\Omega_2}^{n+1}$.

7. Update the level set one time step, as described in Section 4.5, obtaining ϕ^{n+1}.

8. Reconstruct the properties in the two-fluid domain according to the sign of the level set

$$\mathbf{U}_{i,j}^{n+1} = \begin{cases} \mathbf{U}_{\Omega_1,i,j}^{n+1} & \text{if } \phi_{i,j}^{n+1} \leq 0 \\ \mathbf{U}_{\Omega_2,i,j}^{n+1} & \text{if } \phi_{i,j}^{n+1} > 0. \end{cases} \tag{4.34}$$

This completes the time-step.

4.7 Boundary conditions

The NLAA boundary condition is applied on Γ by updating boundary cells using the characteristic boundary condition formalism as described in Chapter 3, with a first-order Euler method for time integration. The value of p_∞ is set locally on each boundary cell to the initial pressure in that cell. Whilst there is no boundary in the physical domain on the polar axis, boundary conditions must be applied here, and at the origin, due to the symmetry imposed on the computational domain. Ghost cells are used to apply reflecting boundary conditions, allowing transverse velocities, along the polar axis boundary and at the origin.

4.8 Benchmarking

Whilst there are many possible test problems for multiphase finite volume schemes in one-dimensional Cartesian coordinate systems, there are a limited number of spherical test problems. In this section I present results of the code on various test cases. Firstly, the one-dimensional Cartesian problem of the Sod shock tube discussed in the previous section. Next I test the code with a spherically symmetric underwater explosion problem, which has been extensively reported in the literature. In two dimensions I test the code on a bubble in equilibrium with the water.

4.8.1 Sod shock tube

In Figures 4.1 and 4.2 in Section 4.6 I show that the code gives good results for simple test problems in a Cartesian coordinate system. I numerically intergate the density, momentum and energy over the domain for the solution to the Sod shock tube problem at $t = 0.2$, for a range of values of δr. By comparing this with the analytical solution to the problem, I obtain global conservation errors. These are shown in Table 4.1. For $\delta r = 0.02$ errors associated with overheating at the interface due to the GFM extend far enough from the interface that they influence the rarefaction wave, and hence the case of $\delta r = 0.02$ does not follow the trends of finer meshes. For $\delta r \leq 0.01$ the global conservation errors for mass and energy are proportional to δr. The momentum errors show a weak convergence ($E_{\text{momentum}} \propto \delta r^n$ with $n < 1$) for $\delta r \leq 2.5 \times 10^{-3}$.

δr (m)	E_{mass} (%)	E_{momentum} (%)	E_{energy} (%)
0.02	1.184	3.986	0.1564
0.01	−1.069	−0.01849	−1.980
5×10^{-3}	−0.5462	0.1076	−0.9927
2.5×10^{-3}	−0.2542	0.1493	−0.4764
1.25×10^{-3}	−0.1472	0.1079	−0.2710
6.25×10^{-4}	−0.07269	0.06100	−0.1302
2.5×10^{-4}	−0.01850	0.05863	−0.04203

Table 4.1: Percentage errors in global mass, momentum and energy conservation for the Sod shock tube problem with grid refinement.

Figure 4.3 shows the density in the region around the interface (left) and the shock wave (right) for the Sod shock tube problem, for different degrees of mesh refinement. For the contact discontinuity, the jump in density is as sharp as can be supported on the mesh. To the left of the interface there is overheating for all values of $\delta r \leq 0.01$, and the extent of this is proportional to δr. For $\delta r = 0.02$ the overheating is reduced, although the density to the right of the interface is incorrect. For the shock wave, there is smoothing for the results with all values of δr, and the extent of the smoothing is

proportional to δr. Again, for the case of $\delta r = 0.02$ the strength and position of the shock are incorrect. The different behaviour for $\delta r = 0.02$ can be explained, as above, by the fact that the overheating errors about the interface extend over a large enough region to interact with the shock and rarefaction waves. These errors are influenced by the initial conditions.

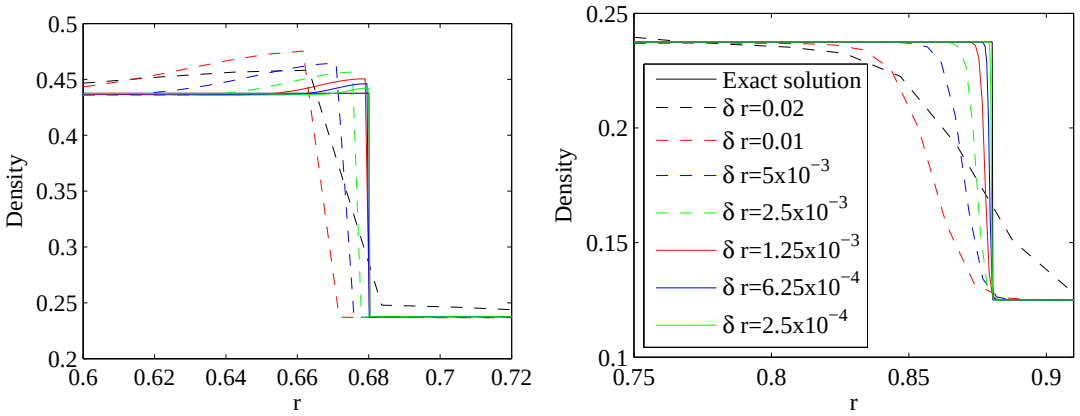


Figure 4.3: Spatial density profiles for the Sod shock tube problem at time $t = 0.2$, about the contact discontinuity (left) and the shock wave (right) for various values of δr. The solid black trace corresponds to the exact solution

4.8.2 Spherically symmetric underwater explosion

The second test problem I use for benchmarking is the underwater explosion problem, mentioned in Section 4.6. I investigate this problem more extensively in Chapter 5. The initial conditions are the same as for the air-water shock tube problem in Section 4.6, but with the imposition of spherical symmetry, and the interface initially located at $r = 0.16$m. They are

$$(\rho,\, u,\, p,\, \gamma,\, p_c) = \begin{cases} (1.63,\, 0,\, 83810,\, 1.4,\, 0) & \text{if } 0 \leq r \leq 0.16, \\ (1.025,\, 0,\, 10,\, 5.5,\, 4921.15) & \text{if } 0.16 < r \leq R_D. \end{cases} \tag{4.35}$$

Note again that these initial conditions have been dimensionally scaled, such that density has units of grammes per cubic centimeter, pressure has units of Bar (10^5Pa) and velocity has units of 10ms^{-1}. This problem was first introduced by Flores and Holt

(1981), and has since been well studied in the literature, for example, by Cocchi *et al.* (1996), Wardlaw and Mair (1998), Smith (1999), Liu *et al.* (2001a), Luo *et al.* (2004), Hu *et al.* (2006), Pishevar and Amirifar (2010) and Barras *et al.* (2012). Table 4.2 shows the maximum bubble radius $R_{int,\max}$ and the bubble period t_{collapse} for this problem as δr is varied. In these simulations $R_D = 4$m. Both aspects of the bubble behaviour show show a first order convergence rate as δr is decreased. This is in agreement with the first order convergence as δr is decreased, later observed for the air gun bubble problem in Table 5.2. Taking values from Hu *et al.* (2006) of $R_{int,\max} = 3.23$m and $t_{\text{collapse}} 1.96 \times 10^{-1}$s, Table 4.2 shows 0.99% and 1.02% errors in maximum bubble radius and bubble period respectively for $\delta r = 1.25 \times 10^{-3}$.

δr (m)	$R_{int,\max}$ (m)	t_{collapse} (10^{-1}s)
0.02	3.097	1.8666
0.01	3.173	1.9418
5×10^{-3}	3.209	1.9376
2.5×10^{-3}	3.227	1.9484

Table 4.2: The maximum bubble radius and collapse time for the underwater explosion problem as δr is varied.

Whilst I leave a detailed discussion of the early stages of the underwater explosion to Chapter 5, I note that during the first millisecond, the positions of shocks and pressure waves within the bubble in my model match those published in Hu *et al.* (2006) to within 3.5%, for $\delta r = 2.5 \times 10^{-3}$m.

4.8.3 Two-dimensional stationary bubble

A fundamental test of the code is to ensure that it does nothing in the correct situations. This two-dimensional test problem consists of an initially stationary bubble at equilibrium with its surroundings, in the absence of gravity. The system is in equilibrium, and therefore the velocities should remain zero and the pressures constant.

The initial conditions are

$$(\rho,\, \mathbf{u},\, p,\, \gamma,\, p_c) = \begin{cases} \big(1.5,\, 0,\, 1.77 \times 10^5,\, 1.4,\, 0\big) & \text{if } 0 \leq r \leq 0.1, \\ \big(1000,\, 0,\, 1.77 \times 10^5,\, 7,\, 3 \times 10^8\big) & \text{if } 0.1 < r \leq R_D. \end{cases} \tag{4.36}$$

All properties are expressed in S.I. units, and gravity is neglected by setting $g = 0$. I set $m = n = 50$ and $R_D = 1$.

When the simulation is run under these initial conditions, there is no resulting motion. The boundary condition is stable in equilibrium, which is consistent with its description in equations 3.37 to 3.42.

4.9 Limitations of the numerical scheme

The numerical scheme is first-order in both space and time. I have investigated using higher-order schemes including a second-order MUSCL scheme (Colella, 1985) and a 5th-order WENO scheme (Borges *et al.*, 2008) with third-order time integration (Shu and Osher, 1988). I find that the first-order scheme provides best results. I compare my results for a one-dimensional underwater explosion problem with the results of other authors (Hu *et al.* (2006) and Luo *et al.* (2004)). The first order scheme provides results which closely match those of Hu *et al.* (2006) and Luo *et al.* (2004), as is shown in Chapter 5, Sections 5.5 and 5.6, whilst higher-order methods lead to severely damped bubble oscillations. I believe this is due to the different momentum and energy fluxes through the interface when the GFM is used in conjunction with a numerical scheme based on a wide stencil.

A flaw in the current scheme is that the Euler equations in polar coordinates are not in conservative form, due to the geometric source terms. When the motion of the interface is in the radial direction, the regions either side of the interface are subject to erroneously high or low energies, and the Rankine-Hugoniot conditions are not met at the interface. The obvious symptom of these errors is a pressure discontinuity at the interface, which is proportional in magnitude to the radial interface speed and the grid

size, and inversely proportional to the distance of the interface from the origin. This error is reduced by refining the computational mesh. This is an open problem, which I discuss further in Appendix B.

The NLAA boundary condition is designed around a polar coordinate system. Polar coordinate systems contain a singularity at the origin and also along the polar axis. This causes problems for the code when run in two dimensions, and eventually leads to the breakdown of the code. These problems are discussed further in Chapter 6. In a cylindrical coordinate system the singularity at the origin is removed, and the singularity along the polar axis is weakened. The NLAA boundary condition may be implemented in a scheme based on a cylindrical coordinate system, on a domain defined as the set of points $\mathbf{r} = (r_{\text{cyl}}, z) \in ([0, R_D], [-R_D, R_D])$. This can be done by rotating the velocities and spatial derivatives on the boundary to obtain the components aligned with the radial and polar directions. The boundary condition may then be applied as described in Chapter 3, then the velocities are transformed back into components aligned with the cylindrical coordinate system. However, I find this method to be unstable, producing waves which travel back and forth along the boundaries, growing in amplitude until the simulation breaks down. When the boundary condition is applied in this manner the effective domain boundary becomes jagged in regions where the vector normal to the boundary is not aligned with $\mathbf{e_r}$. This leads to approximations being made regarding the domain of dependance of the solution on the boundary. Hence parts of the solution which are in fact propagating out of the domain may be inadvertantly propagated along the domain boundary.

Due to the regular mesh, the cells near the origin are very small for the two-dimensional simulations, and the constraining cell when determining the value of the time step is usually near the origin, despite the much larger sound speeds in water away from the origin. A different meshing technique would remove this limitation, and could provide results at a lower computational cost. For the two-dimensional model, with $m = n = 50$, approximately 3×10^4 time steps are required to simulate one bubble oscillation. Neglecting the time associated with the initialisation and termination of the code, for $m = n = 50$, the code takes approximately 20ms per time step on an ordinary

desktop computer. By varying the number of computational cells, but maintaining $m = n$, I find that the computational cost per time step is approximately proportional to $(1 + 0.02m)\, m^2$.

In Section 4.8 I demonstrate that there are errors due to the discretisation, which are reduced with mesh refinement. It is necessary to find a compromise between accuracy and computational cost. For the two-dimensional results, I find that adequate results can be obtained with $m = n = 50$, and in general, run the simulations with these values. In certain cases in Chapters 6, 7 and 8 I present results obtained using a finer mesh.

Chapter 5

One-dimensional results

5.1 Introduction

In this chapter I present one-dimensional results obtained using the numerical scheme described in Chapters 3 and 4. I test the method on both one- and two-phase test problems. The results demonstrate and quantify the efficacy of the boundary condition developed in Chapter 3. In Section 5.2 I show the results of a single phase test problem. In Sections 5.3 and 5.4 I show results of a one-dimensional simulation of an air gun bubble. Sections 5.5 and 5.6 contain results for an underwater explosion problem. Section 5.7 is a summary of conclusions.

For one-dimensional test problems gravity is neglected. In all cases the largest value of R_D is chosen such that there is insufficient time during the simulation for errors caused by the boundary condition to propagate back into the region of interest. This provides what is effectively an 'ideal' boundary condition, against which to test the NLAA boundary condition. I refer to these cases as 'large domain' or 'ideal' boundary condition cases. In all cases, the computational domain is defined by the set of points $r \in [0, R_D]$, and is made up of uniform cells of width δr, and subject to spherical

symmetry. Test problems are run with a range of values for R_D and δr. All cases are run with a CFL number of 0.8.

5.2 Problem I - Travelling pulse in water

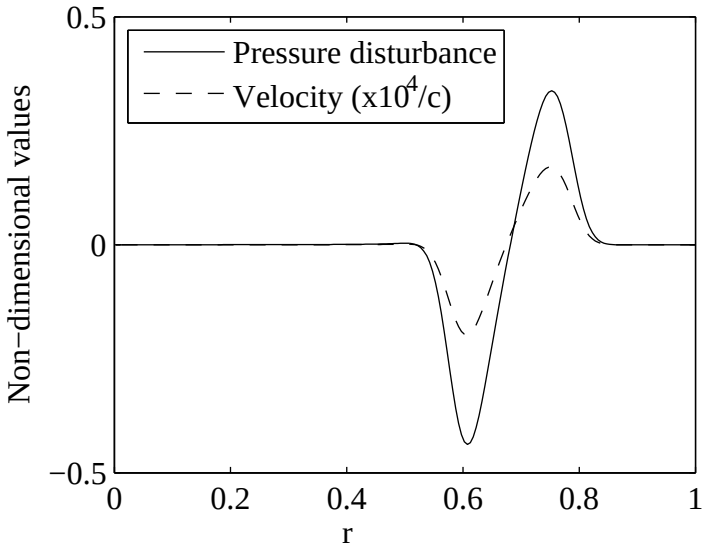


Figure 5.1: Problem I - Spatial pressure disturbance and velocity profiles for the outgoing pulse at time $t = 5 \times 10^{-4}$ seconds. Note that the velocity profile has been made non-dimensional with the local speed of sound and magnified by a factor of 10^4.

I first consider a single phase problem. The problem consists of a domain containing water initially at rest, with uniform density. The pressure of a sphere of water near the origin is increased relative to the surrounding water. These initial conditions produce a pulse which propagates outwards at the local speed of sound. Behind the outgoing pulse the velocity is zero and the pressure is uniform. The initial conditions are

$$(\rho,\, u,\, p,\, \gamma,\, p_c) = \begin{cases} (1,\, 0,\, 10,\, 7,\, 3000) & \text{if } 0 \leq r \leq 0.1, \\ (1,\, 0,\, 1,\, 7,\, 3000) & \text{if } 0.1 < r \leq R_D. \end{cases} \tag{5.1}$$

Note that the dimensions have been scaled such that density, pressure and velocity have units of grammes per cubic centimeter, Bar ($\times 10^5$Pa) and 10ms^{-1}, respectively. The simulation is run on a grid with $\delta r = 0.005$ for 400 time steps. I run the simulation on

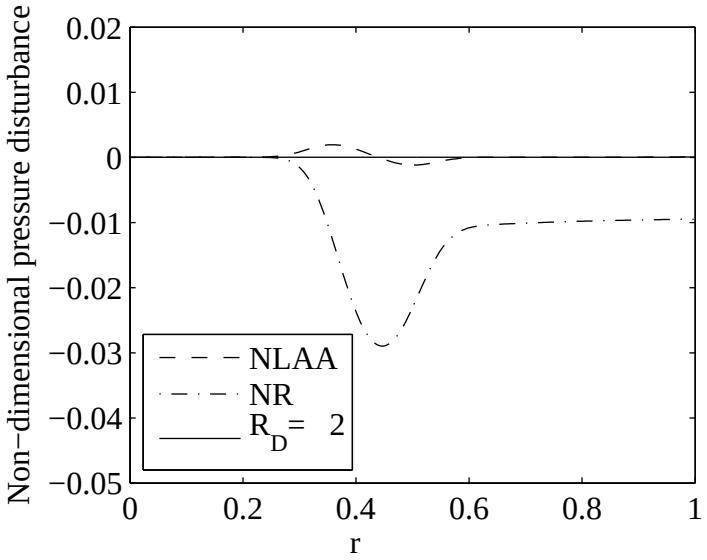


Figure 5.2: Problem I - The remaining pressure disturbances after the main pulse has left the domain due to different boundary conditions, at time $t = 11 \times 10^{-3}$ seconds. Solid line: large domain 'ideal' boundary condition. Dashed line: non-linear acoustic approximation boundary condition. Dot-dashed: Thompson's (1990) non-reflecting boundary condition.

a domain with size $R_D = 1$ for my new artificial boundary condition (NLAA) and also for Thompson's (1990) non-reflecting boundary condition (NR). I run the simulation on a domain with $R_D = 2$ to provide an ideal boundary condition. I calculate the pressure disturbance as the relative deviation of the absolute pressure from the initial pressure at the domain boundary (ie. $(p(r,t) - p_\infty)/p_\infty$, and because in this case $p_\infty = 1$, $(p(r,t) - p_\infty)/p_\infty = p(r,t) - 1$). I make the velocity non-dimensional by multiplying by $10^4/c$, where c is the local speed of sound. Figure 5.1 shows the velocity and pressure disturbances due to the outgoing pulse. Figure 5.2 shows pressure disturbances, caused by the artificial boundary condition, propagating back towards the origin. It is apparent from Figure 5.2 that in this case the NLAA boundary condition outperforms the NR boundary condition: the disturbance which propagates inwards is of much smaller magnitude than that produced by Thompson's (1990) boundary condition. The final pressure reached after a long time has elapsed is correct when using the NLAA boundary condition, but not when using the NR boundary condition. In this case the Mach number of the pulse as it impacts on the domain boundary is very low, at 1.5×10^{-5}.

The strength of the pulse leaving the domain is very weak, and this case satisfies the assumptions on which the non-linear acoustic approximation is based. The amplitude of the spurious pulse caused by the boundary condition is approximately 350 times smaller than the amplitude of the pulse which propagates outwards.

5.3 Problem II-A.1 - Air gun bubble - early stages

I consider the problem of the bubble produced by a seismic air gun. This problem consists of an initially stationary bubble of air at high pressure in water. The initial conditions are

$$(\rho,\, u,\, p,\, \gamma,\, p_c) = \begin{cases} \big(102,\, 0,\, 8.85 \times 10^6,\, 1.4,\, 0\big) & \text{if } 0 \leq r \leq 0.1, \\ \big(1000,\, 0,\, p_w,\, 7.0,\, 3 \times 10^8\big) & \text{if } 0.1 < r \leq R_D, \end{cases} \tag{5.2}$$

where $p_w = p_{atm} + 1000 \times 9.81 \times 7.7$ and $p_{atm} = 1.01325 \times 10^5$. All properties are expressed in S.I. units. This problem is equivalent to an air gun with a volume of 250 cubic inches, charged to a pressure of 2000 pounds per square inch at a depth of 7.7 metres. Air guns of this size and pressure are commonly used in industry. Note that the discrepancy in initial pressures - 8.85×10^6Pa $\approx$ 1300psi - is intentional, and is designed to account for the process by which air is released from the gun. I run the simulation for domain sizes R_D, of 1 and 5 metres. In both simulations, a grid cell size of $\delta r = 2 \times 10^{-4}$metres is used. The results with $R_D = 5$ are taken to be the ideal boundary condition case.

Figure 5.3 shows the pressure profiles at different times for the two domain sizes. Note the small discontinuity in pressure at the interface due to the non-conservative form of the Euler equations in polar coordinates, discussed in Chapter 4 and Appendix B. For both domain sizes, the pressure profiles match very closely and cannot be distinguished in Figure 5.3. As the outgoing pressure wave passes the domain boundary a disturbance due to the artificial boundary condition forms and propagates back into the domain. This disturbance causes density, velocity and pressure errors at the boundary of -3×10^{-5}%, 0.007% and -0.04% respectively.

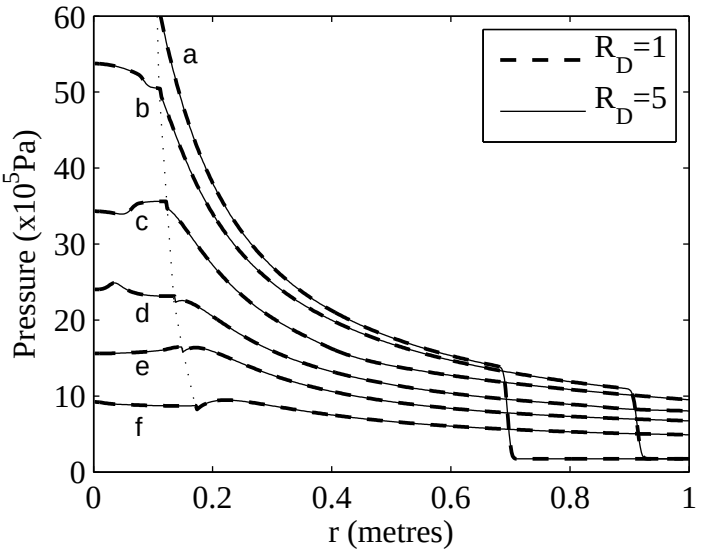


Figure 5.3: Problem II-A.1: Spatial pressure profiles at different times for domain sizes of $R_D = 1$ (NLAA BC) and $R_D = 5$ ('ideal' BC). (a) $t = 0.22$ms; (b) $t = 0.29$ms; (c) $t = 0.43$ms; (d) $t = 0.59$ms; (e) $t = 0.75$ms; (f) $t = 1.10$ms. The dotted line shows the location of the interface between air and water.

5.4 Problem II-A.2 - Air gun bubble - long run

I now consider the same problem as in the previous section but on longer time scales. I run the simulation for domain sizes R_D, of 1, 2, 4, 8, 16 and 125 metres, with a cell size of $\delta r = 5 \times 10^{-3}$ metres. The simulation is run for 5×10^4 time steps, which corresponds to approximately 0.14 seconds, during which time the bubble undergoes two full oscillations. During the simulation, the maximum outgoing pressure wave impacts on the boundary in the $R_D = 125$ case; there is insufficient time for any disturbances to propagate back towards the origin as far as $r = 16$. I take the case of $R_D = 125$ as the case with ideal boundary condition with which to compare results obtained on smaller domains.

Figure 5.4 shows the time evolution of the interface position, R_{int}, and pressure, P_{int}, for $R_D = 1$, 2 and 125. The cases of $R_D = 4$, 8 and 16 are omitted from Figure 5.4 as the match with the case of $R_D = 125$ is so close as to be indistinguishable by eye. Figure 5.5 shows the magnitude of the relative error of the maximum interface position

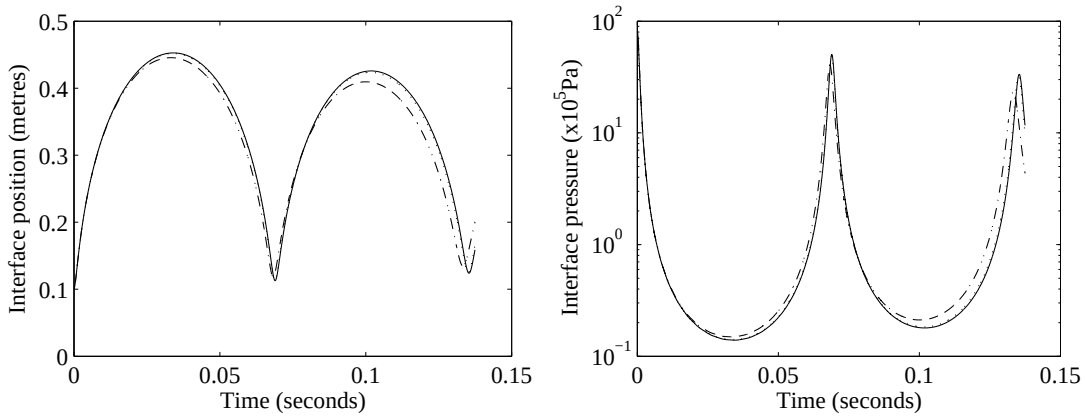


Figure 5.4: Problem II-A.2: Interface position and interface pressure variation as a function of time for different values of R_D: $R_D = 1$ - dash-dot line; $R_D = 2$ - dotted line; $R_D = 125$ - solid line.

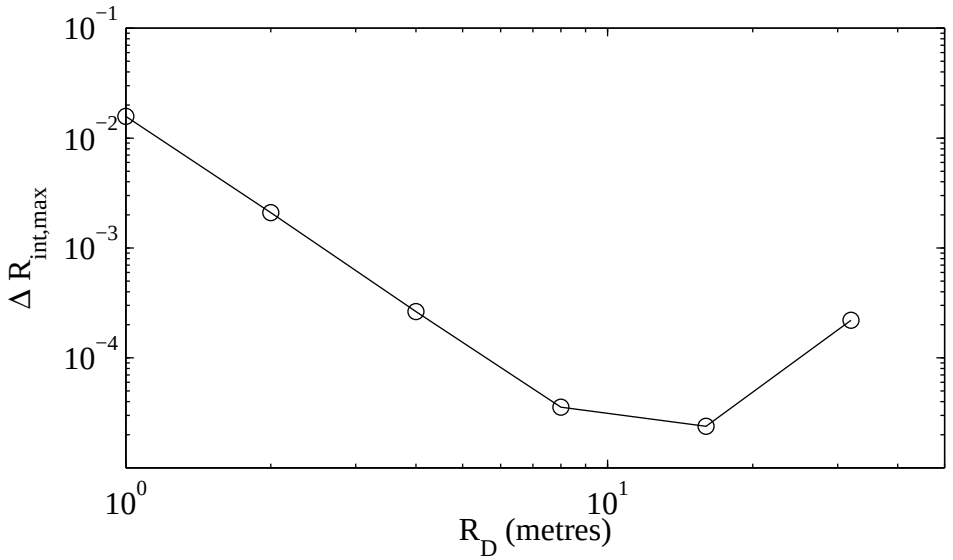


Figure 5.5: Problem II-A.2: Variation of the magnitude of the relative error in maximum interface position for different domain sizes. $\Delta R_{int,max} = |\max(R_{int,R_D}) - \max(R_{int,R_D=125})| \,/\, \max(R_{int,R_D=125})$.

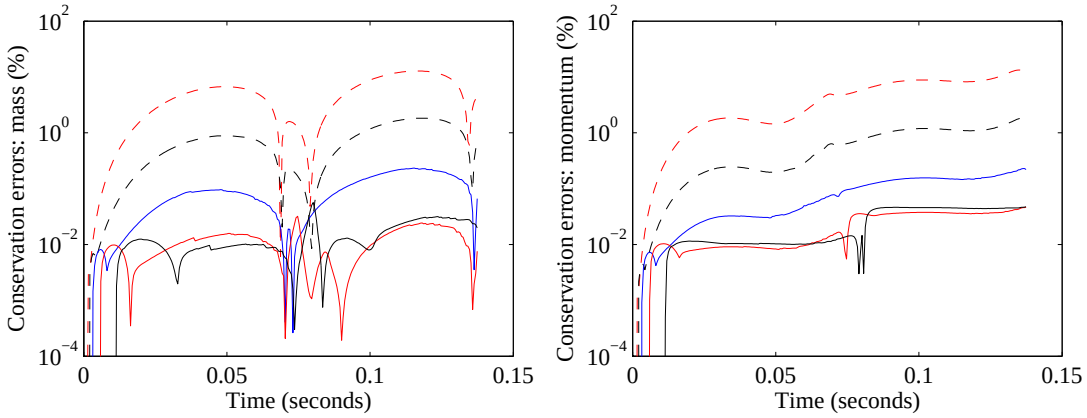


Figure 5.6: Problem II-A.2: Relative mass and momentum boundary conservation errors as functions of time for different values of R_D: $R_D = 1$ - dashed red line; $R_D = 2$ - dashed black line; $R_D = 4$ - solid blue line; $R_D = 8$ - solid red line; $R_D = 16$ - solid black line.

as R_D is varied. The results show third-order convergence of R_{int} with increasing R_D. P_{int} also shows third-order convergence. The convergence rate fails for $R_D = 16$ and 32, as the variation in density on R_D in these cases is of the same order of magnitude as machine precision errors.

As mentioned above, it can be seen in Figure 5.4 that as R_D is changed, the damping and period of the bubble oscillations also changes. For the case of $R_D = 125$, the boundary is far enough from the origin that the initial outgoing pressure pulse does not have sufficient time to interact with the boundary and return to the bubble. Therefore for this case, the damping and period of the bubble must be unaffected by the boundary condition. It is worth considering the causes of the damping for this case. When the boundary is far enough from the bubble to have no influence, observed damping may be caused by (1) dissipation in the GFM, (2) numerical viscosity inherent in the single-phase Euler solver at the heart of the code and (3) the compressibility of the water.

It is shown in the following subsection that the choice of GFM influences bubble period and damping. However, the influence of the GFM is reduced with mesh refinement. Furthermore, given that the code converges on "correct" results in Section 4.8 in Chapter 4, this suggests that for grid converged results the damping is not caused by the GFM. The numerical viscosity inherent in the scheme is unavoidable, although

it is reduced with the higher order spatial discretisation and time integration schemes. However, as mentioned in Chapter 4, higher order schemes interact with the GFM differently, producing incorrect results.

The compressibility of the water causes damping as the energy of motion is converted to internal energy in the water as it is compressed. To investigate this damping mechanism, I define a measure of the damping of the bubble ξ_b as the ratio of the initial pressure in the bubble to the peak pressure at the interface during the first collapse. An increase in the value of ξ_b corresponds to an increase in damping. The speed of sound in water (as governed by the stiffened gas equation of state) is given by $c = \sqrt{\gamma\left(p + p_c\right)/\rho}$. The isentropic compressibility of water is given by $\beta_S = 1/\left(\rho c^2\right)$. For this analysis, $p_c \gg p$, and therefore I neglect p, allowing β_S to be simplified to $\beta_S = 1/\left(\gamma p_c\right)$. For $\gamma = 7$ and $p_c = 3 \times 10^8$Pa, $\beta_S = 4.762 \times 10^{-10}\text{Pa}^{-1}$.

I run the simulation with $R_D = 4$ for a range of values of γ and p_c, and in each case calculate β_S and ξ_b. The results are plotted in Figure 5.7. Clearly there is a trend for an increase in damping with increased compressibility. Furthermore, some of the data points are obtained by adjusting γ, some by adjusting p_c, and some by adjusting both. Despite this, all the data points follow the same trend. In an incompressible fluid, inviscid fluid, the oscillations should be undamped, with $\xi_b = 1$. The trace in Figure 5.7 appears to tend towards a value of $\xi_b > 1$ as β_S tends to zero. This can be attributed to the damping caused by the GFM at this resolution, the numerical viscosity inherent in the scheme, and the damping due to the boundary condition.

For each value of R_D, the fluxes of the conservative properties are calculated at the domain boundary. The fluxes are also calculated at the same position for the case of $R_D = 125$. These fluxes are then integrated with respect to time to determine the total quantity of each conserved property which has left the domain. I then determine the relative errors in these cumulative fluxes, taking the case of $R_D = 125$ as a reference. Figure 5.6 shows the relative errors in the conservation properties of the boundary for differing domain sizes. Boundary conservation errors in energy match those in mass to within 0.1% in all cases. Figure 5.6 shows a maximum error in boundary

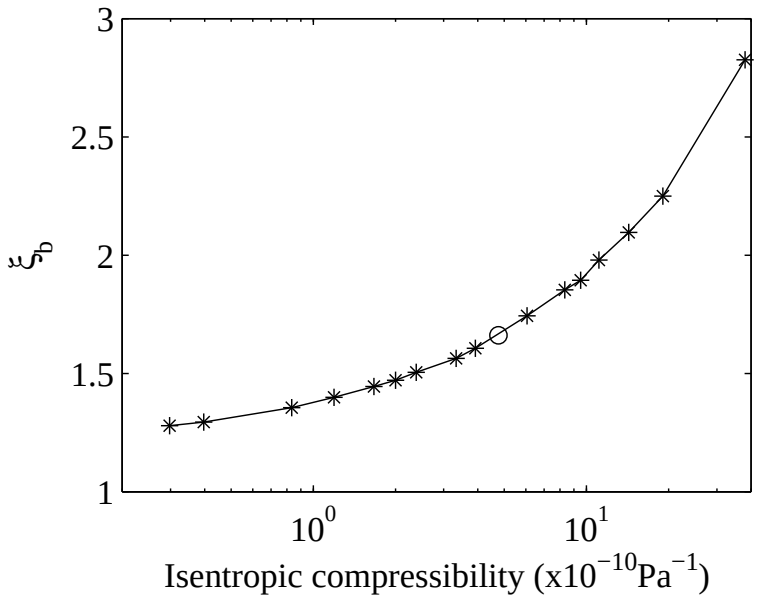


Figure 5.7: Problem II-A.2: Bubble damping as a function of isentropic compressibility of the water around the bubble. The point marked with a circle corresponds to the true value of compressibility of water.

conservation after two full bubble oscillations (5×10^4 time steps) of approximately 10%. Figure 5.6 also shows a third-order convergence in boundary conservation errors as R_D is increased. When $R_D = 16$, relative variation in density at the domain boundary is of the same order of magnitude as machine precision errors, and hence the convergence characteristics of the smaller domains in Figure 5.6 appear not to hold.

This convergence rate is independent of mesh size for $\delta r < 0.01$, although errors due to the boundary conditions are reduced with finer mesh. If $\delta r > 0.01$ the errors associated with the interface mentioned in the previous section impinge on the boundary, and the convergence rate of errors with increasing R_D drops to second-order. I run the simulation for two domain sizes, $R_D = 1$ and $R_D = 2$ for a range of grid sizes, $\delta r = 5 \times 10^{-3}$, 2×10^{-3}, 1×10^{-3} and 3.33×10^{-3} metres, corresponding to 200, 500, 1000 and 3000 cells per metre respectively. With 3000 cells per metre, the simulation was run for 5×10^5 time steps. The relative error in interface position, $E_{R_{int}}$, and the relative error in interface pressure $E_{P_{int}}$ between the two domain sizes was calculated for each grid size. Table 5.1 shows the L_1 norm and the convergence rates for these errors.

δr (metres)	$E_{R_{int}}$	R_c	$E_{P_{int}}$	R_c
5×10^{-3}	7.05×10^{-3}		0.112	
2×10^{-3}	3.08×10^{-3}	0.916	0.047	0.958
1×10^{-3}	1.57×10^{-3}	0.981	0.024	0.991
3.33×10^{-4}	5.53×10^{-4}	0.946	8.3×10^{-3}	0.947

Table 5.1: Problem II-A.2: Relative errors due to boundary conditions and convergence rate with mesh refinement.

δr	0.02	0.01	5×10^{-3}	2.5×10^{-3}	1.25×10^{-3}	6.25×10^{-4}
$R_{int,\max}$	0.4123	0.4347	0.4468	0.4534	0.4570	0.4589

Table 5.2: Problem II-A.2: Variation of maximum bubble radius with mesh refinement.

I find approximately first-order convergence of these errors with grid size. Table 5.2 shows the maximum bubble radius for various values of δr, with $R_D = 1$. Again there is first order convergence of $R_{int,\max}$ as δr is decreased.

5.4.1 Influence of GFM on bubble oscillation

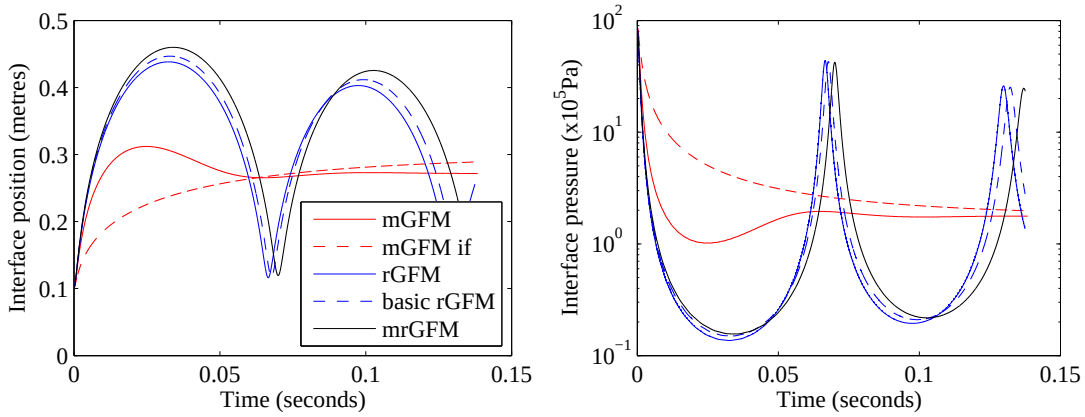


Figure 5.8: Problem II-A.2: Variation of interface position and pressure with time in a one-dimensional model for various versions of the ghost fluid method.

Figure 5.8 shows the variation of R_{int} and P_{int} with time obtained with the various

versions of ghost fluid method described in Chapter 4. It is clear that whilst the initial motion is very similar for all ghost fluid methods tested, as the interface moves further the results differ significantly. The versions of GFM based on Riemann solvers show roughly similar oscillatory behaviour, whilst the GFMs based on extrapolation show overly damped bubble motion. It is clear that the choice of ghost fluid method has a major effect on the overall results of the model. This is to be expected, as the overall bubble motion is determined by the rate of transmission of momentum and energy between the air and the water, and the mechanism which controls this transmission of momentum and energy is the ghost fluid method. Whilst the mGFM may provide accurate results for the shock tube problems above, the early stages of bubble oscillation, and the test cases in Fedkiw (2002) and Liu *et al.* (2003) with shocks impacting on material interfaces, as the interface moves across many computational cells, large errors accumulate. It is interesting to note that when the basic rGFM is used in conjunction with a higher order numerical scheme (eg. MUSCL), the results are similar to the first-order scheme with the mGFM.

The differences between the rGFM, the basic rGFM and the mrGFM are clearly visible in Figure 5.8, although all three show qualitatively similar behaviour. The rGFM gives the smallest maximum interface position, with a shorter period and least damping on collapse. The mrGFM gives the biggest maximum interface position, with more damping as the bubble collapses and a longer period. The basic rGFM provides results which are between the rGFM and the mrGFM. Interestingly, despite the mrGFM giving the largest maximum bubble radius, it also gives the largest minimum pressure. These differences are due to the accumulation of small differences in conservation properties of the different GFMs over the long run time. The purpose of this research is to demonstrate the artificial boundary condition using the NLAA, and to demonstrate that the present model can qualitatively capture the behaviour of air gun bubbles. Hence, the choice of GFM is not crucial, and any of the three versions of rGFM are adequate.

5.5 Problem II-B.1 - Gaseous explosion in water - early stages

I consider an underwater explosion problem first studied by Flores and Holt (1981). Other authors have investigated this problem, using both Eulerian (Cocchi *et al.* (1996), Liu *et al.* (2001a) and Hu *et al.* (2006)) and arbitrary Lagrangian-Eulerian (ALE) (Wardlaw and Mair (1998), Smith (1999), Luo *et al.* (2004), Pishevar and Amirifar (2010) and Barras *et al.* (2012)) methods. This problem is similar to the problem of modelling a seismic air gun, but with a much greater initial pressure. The strength of the propagating shock in this problem is greater than the non-linear acoustic approximation was designed for. As this problem is beyond the remit of the NLAA, it is a good test for the robustness of my method. The initial conditions are

$$(\rho,\, u,\, p,\, \gamma,\, p_c) = \begin{cases} (1.63,\, 0,\, 83810,\, 1.4,\, 0) & \text{if } 0 \leq r \leq 0.16, \\ (1.025,\, 0,\, 10,\, 5.5,\, 4921.15) & \text{if } 0.16 < r \leq R_D. \end{cases} \tag{5.3}$$

Note that these initial conditions have been dimensionally scaled, such that density has units of grammes per cubic centimeter, pressure has units of Bar (10^5Pa) and velocity has units of 10ms^{-1}. The simulation is run for domain sizes of $R_D = 1$ and $R_D = 5$, with $\delta r = 3.33 \times 10^{-4}$ metres in both cases. The case of $R_D = 5$ provides the ideal boundary conditions. The maximum Mach number which occurs in the bubble is 1.08, and in the water is 0.37. The maximum Mach number at $r = 1$ is 0.11.

Figure 5.9 shows the pressure profile at different times for both $R_D = 1$ and $R_D = 5$. Initially a shock wave propagations are from the interface into the water, and a rarefaction wave propagates into the bubble towards the origin. The shock wave is visible at about $r = 0.7$ in curve (a). As the rarefaction wave impacts on the origin it is reflected as a rarefaction wave, the pressure near the origin drops below the pressure in the rest of the bubble, and an inward propagating shock forms. This shock reflects of the origin and propagates outwards (it is visible at about $r = 0.1$ in curve (a)). Curve (b) shows this shock just prior to impacting on the interface. When it impacts on the interface, it is partially reflected back towards the origin, and partially

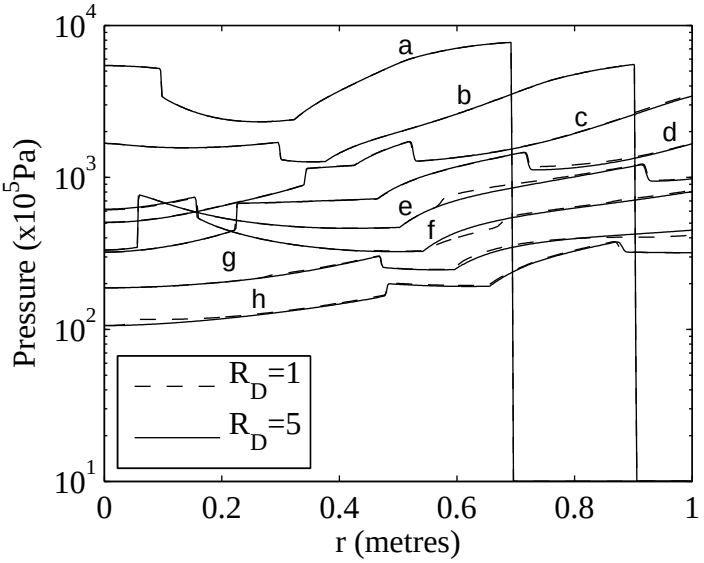


Figure 5.9: Problem II-B.1: Spatial pressure profiles at different times for domain sizes of $R_D = 1$ (NLAA BC) and $R_D = 5$ ('ideal' BC). (a) $t = 0.21$ms; (b) $t = 0.31$ms; (c) $t = 0.41$ms; (d) $t = 0.5$ms; (e) $t = 0.6$ms; (f) $t = 0.73$ms; (g) $t = 0.91$ms; (h) $t = 1.13$ms.

transmitted out into the water (curve (c)). A thorough explanation of this behavour is given by, for example, Wardlaw and Mair (1998). As the outgoing pressure wave impacts on the boundary a small disturbance forms and propagates back towards the origin (curves (c), (d) and (e)). The disturbance causes maximum errors of -15%, 1.5% and -0.1% in the pressure, velocity and density, respectively, at the boundary. The disturbance propagates in to the air-water interface, at which point it is partially reflected outwards, and partially transmitted into the bubble, where it grows in strength as it converges on the origin (curves (f), (g) and (h)). That the pressure disturbance is positive implies that the boundary condition provides too strong an impedance, forcing fluxes through the boundary to be lower than in the ideal case. This is re-inforced by the fact that increasing R_D gives an increase in the maximum value of R_{int}.

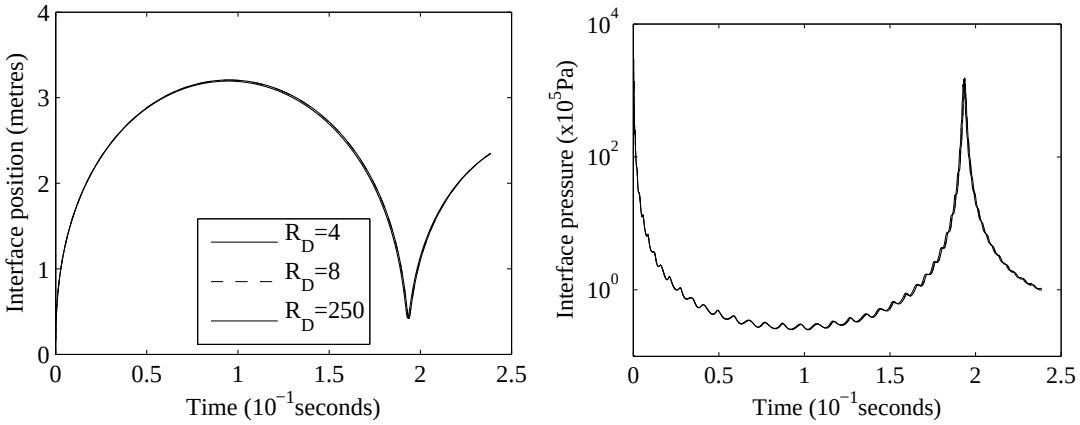


Figure 5.10: Problem II-B.2: Interface position and interface pressure variation as a function of time for different domain sizes.

5.6 Problem II-B.2 - Gaseous explosion in water - long run

I now test the method on the underwater explosion problem over a much greater run time, for one full bubble oscillation period. The initial conditions are the same as for the previous case. I now use a coarser grid, with $\delta r = 4 \times 10^{-3}$ metres. The simulation is run for domain sizes of $R_D = 4, 6, 8$ and 250, for 1.25×10^5 time-steps. During the simulation the outgoing pressure wave reaches the boundary in all cases. When $R_D = 250$ spurious reflections from the boundary do not have time to propagate further inwards than $r = 100$. Hence the case of $R_D = 250$ is taken to be the ideal boundary condition with which to compare the performance of the boundary conditions on the smaller domains. The maximum Mach number at $r = 4$ is 0.02.

Figure 5.10 shows the time-evolution of the bubble radius and the interface pressure. The trace for $R_D = 6$ is omitted for clarity. These results are in good (to within approximately 1%) agreement with previous authors (Pishevar and Amirifar (2010) and Hu *et al.* (2006)). The secondary oscillations ('internal bubble oscillations') present in the interface pressure in Figure 5.10 are due to pressure waves propagating across the bubble and interacting with the air-water interface. The maximum radius of the bubble during the simulation is 3.2 metres. The maximum Mach number at the domain

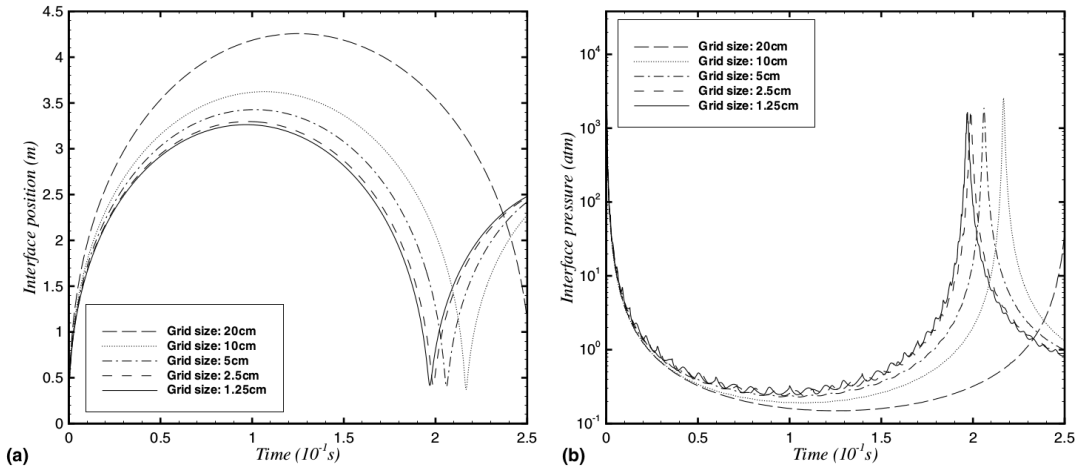


Figure 5.11: Problem II-B.2: Taken from Hu *et al.* (2006), Figure 14. Interface position and interface pressure variation as a function of time for ifferent mesh refinements. Note the good match between Figure 5.10 and the grid-converged results of Hu *et al.* (2006).

boundary when $R_D = 4$ is 0.02, and initial shock which impacts on the boundary has a pressure ratio of 50.

Figure 5.11 shows the same information as Figure 5.10, but is taken from Hu *et al.* (2006). There is a good match between the traces in Figure 5.10 and the trace in Figure 5.11 for the finest mesh. The internal bubble oscillations visible as small undulations in the interface pressure (right panels) match well, demonstrating that the shock speeds in my model and that of Hu *et al.* (2006) are in good agreement throughout the simulation. Hu *et al.* (2006) use a different approach to simulate the air-water interface. It is interesting to note that whilst in my results (see Table 4.2) the maximum value of R_{int} increases towards the grid-converged solution, in the model of Hu *et al.* (2006) the maximum value of R_{int} decreases towards the grid-converged solution. This is a consequence of the different interface modelling techniques used.

Figure 5.12 shows the time evolution of the error in the total mass of air in the bubble. The variation in Figure 5.12 of order 1% is due to the non-conservative properties the scheme at the interface, due to the ghost fluid method, and away from the interface, due to the non-conservative form of the Euler equations in polar coordinates, and is unavoidable given the current numerical scheme, although it can be reduced with mesh

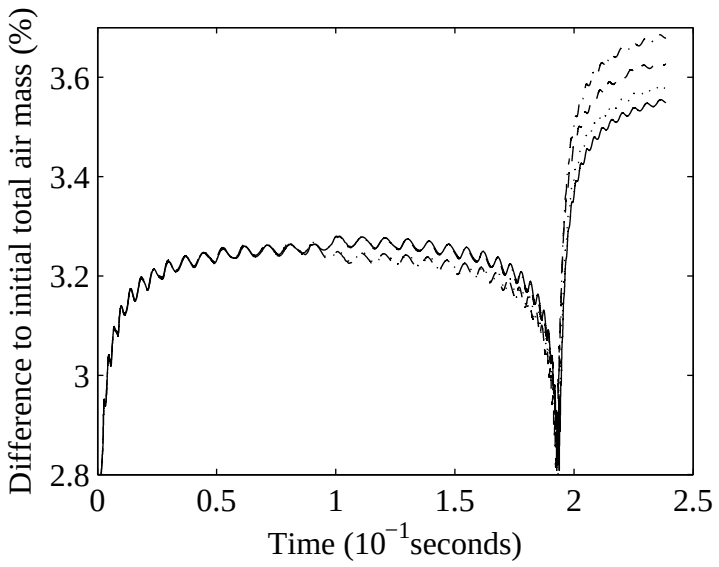


Figure 5.12: Problem II-B.2: Variation of air mass as a function of time for different values of R_D. Dashed line - $R_D = 4$; dash-dot line - $R_D = 6$; dotted line - $R_D = 8$; solid line - $R_D = 250$.

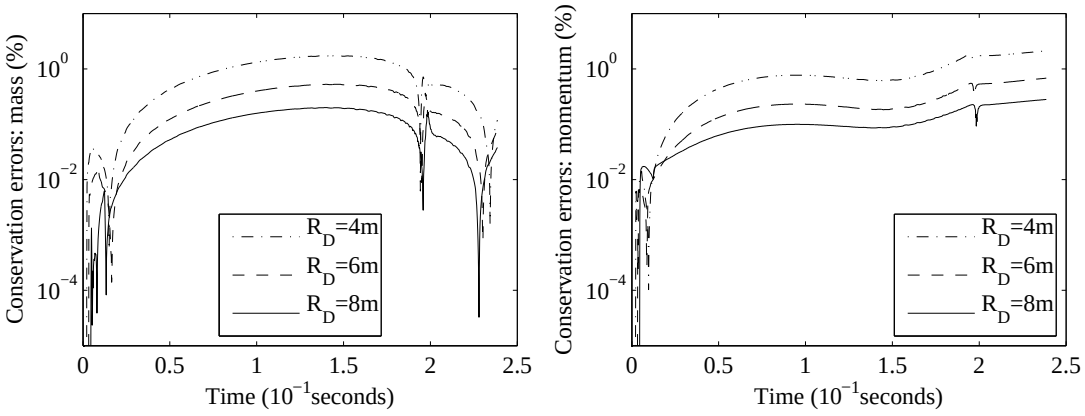


Figure 5.13: Problem II-B.2: Relative mass and momentum boundary conservation errors as a function of time for different values of R_D.

refinement. The differences in mass conservation for different sized domains show no improved performance with larger domains, which suggests that these errors are not caused by the boundary condition, although they may be influenced by it. The variation between the traces in Figure 5.12 is due to the sensitivity of the conservation properties of the scheme to the time at which any disturbances from the domain boundary impact on the material interface. It must be noted that the interaction between the disturbance due to the boundary conditions and the wave inside the bubble has the ability to change the phase of the internal bubble oscillations significantly as the bubble collapses. As with previous test problems, I observe third-order convergence of interface position and pressure as R_D is increased.

Figure 5.13 shows the conservation errors in mass and momentum flowing out of the domain at R_D relative to the $R_D = 250$ case for each value of $R_D = 4$, 6 and 8. As in the previous cases, boundary conservation errors in energy matches those in mass to within 0.1%. These results show a third-order convergence for boundary conservation properties with increasing R_D. For the case of $R_D = 8$, the maximum errors in boundary conservation are of the order of 0.1%. This is a very good conservation property, given that the simulation has been run for such a large number of time steps.

I also run the simulation with $R_D = 3.21$, in which case R_D is 0.1% greater than the maximum bubble radius. In this case the results obtained match those expected from the convergence properties observed above. Recall that the simulation will break down if the interface moves outside the domain. I note that for the underwater explosion problem the minimum acceptable domain size is determined not by the performance of the boundary condition, but by the requirement that the domain is larger than the maximum bubble radius.

The NLAA boundary condition provides excellent results with regard to the large-scale motion of the bubble. Smaller scale motion, such as the pressure waves oscillating within the bubble, are significantly affected by the boundary conditions, but they themselves have little effect on the bubble radius or interface pressure.

5.7 Conclusions

I test the method developed in Chapters 3 and 4 on a range of one- and two-phase problems in one dimension. The method performs well, yielding accurate results for the problems of seismic air gun bubbles and underwater explosions, even when the domain boundary is only slightly larger (0.1%) than the maximum bubble radius. A major benefit of the method is that it allows long run time (10^5 time steps with a CFL number of 0.8) simulations of such problems on a highly truncated domain, at a reduced computational cost. The method is robust, and capable of yielding good conservation properties (errors of less than 1%) over very long run times.

Chapter 6

Two-dimensional results

6.1 Introduction

In Chapter 5 I demonstrate the effectiveness of the NLAA boundary condition in one dimension. If used only in one dimension, the scheme developed in Chapters 3 and 4 provides limited benefit over Rayleigh-Plesset type models, as the imposition of spherical symmetry precludes the model from capturing many interesting phenomena which are inherently asymmetrical. When the NLAA boundary condition is used in two-dimensional simulations, interesting effects such as bubble deformation, the formation of jets due to gravity and surface instabilities may be investigated. In this chapter I present results and discussion of simulations of an air gun bubble in two dimensions using the scheme developed in Chapters 3 and 4. The purpose of this chapter is two-fold. Firstly, the results shown here demonstrate that the boundary condition described in Chapter 3 is not limited to one dimension, and allow a degree of quantification of the conditions in which it may be used. Secondly, I discuss some of the various phenomena mentioned above which are caputured by the two-dimensional model. This discussion also leads to a consideration of the merits and limitations of the present numerical scheme.

Most previous air gun bubble modelling has been one-dimensional - in fact, homogeneous spherical bubbles are typically described entirely by two time-varying parameters with no spatial discretisation - as discussed in Chapter 1. Part of the reason for this is that interest has been in the propagated wavefield in the water, which can be reasonably well determined by considering bubbles to be spherical point sources. The review paper of Cox *et al.* (2004) describes two three-dimensional models of an air gun bubble, although the contents of the bubble are assumed to be homogeneous. Cox *et al.* (2004) is a useful reference, to which I frequently refer throughout this chapter. Two- and three-dimensional simulations of cavitation bubbles (Hu and Khoo, 2004), underwater explosions (Liu *et al.* (2001b), Luo *et al.* (2004), Pishevar and Amirifar (2010), Miller *et al.* (2013) and Zhang *et al.* (2013b)), and even supernovae (Kane *et al.* (2000), Kifonidis *et al.* (2003) and Kifonidis *et al.* (2006)) contain aspects which are relevant to the results presented in this chapter.

The purpose of this work is not to fit a model to experimental data (note, that if this were the case, the model in Appendix D, which was designed to be calibrated with data, would perform much better). The purpose is to develop a model which can capture physical phenomena that existing air gun models can not. For the main part I refrain from drawing quantitative comparisons between my model and earlier air gun models or experimental data. Earlier air gun models are 'fitted' to experimental data with the use of various free parameters and filters with no physical basis. Excluding the reduced pressure in the initial conditions, I omit these fitting parameters from my model. Hence, if I were to draw quantitative comparisons between my model with experimental data, it would appear worse than many previous models. Whilst my approach does not necessarily provide a good match quantitatively, the behaviour of the bubbles is unaffected, and I am able to draw qualitative comparisons between my model, previous models and experimental observations.

Previous air gun modelling has predominantly focussed on the signals emitted by the air gun bubble. A commonly used measure for describing the signal produced by an air gun is the time-variation of pressure at a location one metre from the air gun. In order to draw conclusions from my model regarding the emitted signals, I use an effective bubble

radius (referred to as interface position), and an effective bubble pressure (interface pressure). Changes in these two characteristic variables, such as increased damping, will cause qualitatively similar changes in the pressure wavefield which propagates out into the water.

Section 6.2 contains a description of the initial conditions used for the two-dimensional model. In Section 6.3 I compare the results of the two-dimensional model with those from the one-dimensional model. This section demonstrates the efficacy of the boundary condition. In Section 6.4 I discuss the stability of the bubble surface, the impact of disturbances to the surface on the radiated pressure, and the limitations these impose on the numerical scheme. Section 6.5 contains results showing the deformed shape of the bubble due to gravity, and a discussion of the rate at which the bubble rises. In Section 6.6 I discuss the formation of jets as the bubble collapses, and the problems this phenomenon presents for the present computational scheme. Section 6.7 is a summary of conclusions.

6.2 Initial conditions

Throughout this chapter the initial conditions for the air gun problem are as in Chapter 5, but with the inclusion of hydrostatic pressure terms, p_h and a small disturbance, $\eta = \eta(\theta)$ to the initial bubble radius. The initial conditions are

$$(\rho,\, \mathbf{u},\, p,\, \gamma,\, p_c) = \begin{cases} \left(102,\, 0,\, 8.85 \times 10^6 + p_h,\, 1.4,\, 0\right) & \text{if } 0 \leq r \leq 0.1 + \eta, \\ \left(1000,\, 0,\, p_w,\, 7.0,\, 3 \times 10^8\right) & \text{if } 0.1 + \eta < r \leq R_D, \end{cases} \tag{6.1}$$

where $p_w = p_{atm} + 1000 \times 9.81 \times 7.7 + p_h$, $p_{atm} = 1.01325 \times 10^5$, $p_h = -\rho g r \cos\theta$, and $g = 9.81\text{ms}^{-2}$. The units of density, velocity and pressure are kgm^{-3}, ms^{-1} and Pa, respectively. In those cases where I do not wish to include gravity I simply set $g = 0$. If $g = \eta = 0$, the problem collapses to the one-dimensional problem.

Unless explicity stated otherwise, the computational domain is defined by the set of points $\mathbf{r} \in ([0, R_D], [0, \pi])$, and is split uniformly into cells with side lengths δr and $r\delta\theta$,

where $\delta r = 1/50$, $\delta\theta = \pi/50$ and $R_D = 1$. All simulations in this chapter are run using the modified rGFM described in Chapter 4, Section 4.6. All simulations are run with a CFL number of 0.8.

6.3 Comparison with one-dimensional model

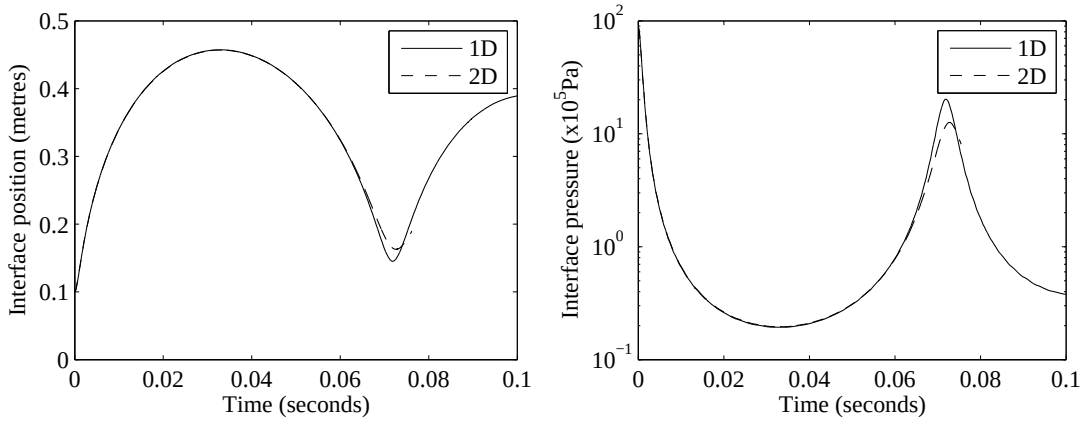


Figure 6.1: Variation of interface position (left) and interface pressure (right) with time for the one- (solid line) and two-dimensional (dashed line) models.

Figure 6.1 shows the interface position and interface pressure for one- and two-dimensional air gun bubble simulations. The two-dimensional model has $g = 9.81$ and $\eta = 0$. Both models are run with $R_D = 1$. For the two-dimensional model, the volume of the bubble is calculated and an effective bubble radius (referred to as the interface position, or R_{int}) is determined as the radius of a sphere with the same volume as the bubble. The interface pressure P_{int}, is calculated by numerically integrating the pressure over the interface to obtain an average.

During the expansion phase ($t = 0$ to $t = 0.035$), the R_{int} and P_{int} obtained with these two models match closely. During the first half of the bubble collapse ($t = 0.035$ to $t = 0.06$) the models continue to give similar results, differing by less than 2%. During the latter stages of collapse ($t = 0.06$ to $t = 0.072$) and the subsequent expansion ($t > 0.072$) the models differ. The two-dimensional model shows increased damping, with a larger minimum volume on collapse, and a reduced peak pressure. The period

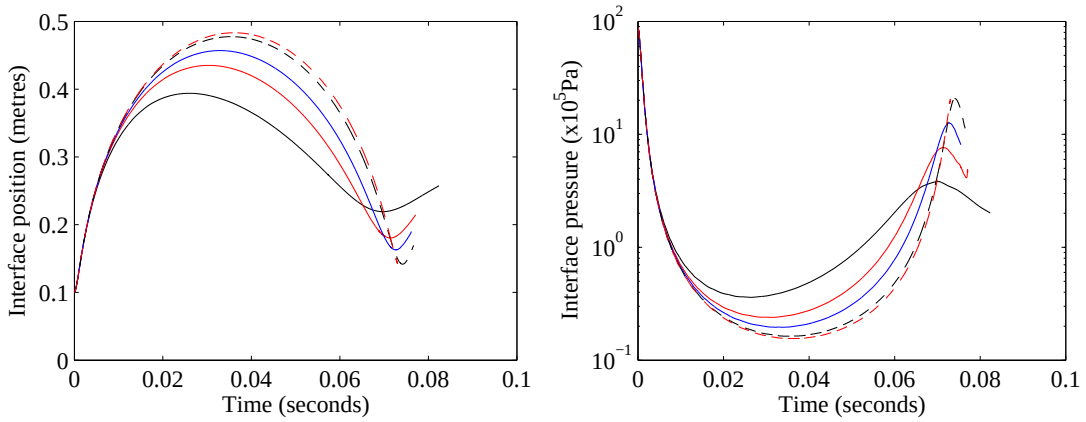


Figure 6.2: Variation of interface position (left) and interface pressure (right) with time for the two-dimensional model for different domain sizes. $R_D = 0.6$ - solid black line; $R_D = 0.8$ - solid red line; $R_D = 1$ - solid blue line; $R_D = 1.5$ - dashed black line; $R_D = 2$ - dashed red line.

of the bubble oscillation is also increased slightly in the two-dimensional model, from 0.072 to 0.073 seconds. These results are to be expected, as the two-dimensional model allows effects such as bubble translation and deformation to occur. These effects present an energy sink, reducing the available energy for oscillation. Shortly after the bubble begins to expand again the two-dimensional simulation breaks down due to numerical instabilities, which are discussed in Section 6.6.

R_D	$\max(R_{int})_{1D}$ $\delta r = 0.02$	$\max(R_{int})_{2D}$ $\delta r = 0.02$	$\max(R_{int})_{2D}$ $\delta r = 5 \times 10^{-3}$
0.6	0.396	0.394	0.435
0.8	0.436	0.435	0.450
1.0	0.457	0.457	0.455
1.5	0.477	0.478	0.459
2.0	0.483	0.483	--

Table 6.1: Maximum bubble radius for different values of R_D.

In its current form, the code would take an inordinately long time (several days) to simulate one oscillation of a two-dimensional bubble on a large domain. Hence I do

not present an 'ideal boundary condition' result to allow the calculation of the absolute errors introduced by the boundary condition in two dimensions.

Figure 6.2 shows the interface position and interface pressure for two-dimensional air gun bubble simulations, with $g = 9.81$ and $\eta = 0$, for domain sizes of $R_D = 0.6$, 0.8, 1, 1.5 and 2 metres. There is a good match between all models in the early stages of expansion. As the interface position approaches a maximum, the results begin to diverge. The results for $R_D = 0.6$ and $R_D = 0.8$ show significantly damped motion. Table 6.1 shows the maximum bubble radius for the one- and two-dimensional models with different domain sizes. Both models show second-order convergence with increasing R_D. This is in disagreement with the results of Chapter 5, where errors showed third-order convergence. The reason for this discrepancy is that, due to the coarse grid used in these simulations, the region near the interface subject to errors due to the polar coordinate system (see Chapter 4 and Appendix B) is large enough that it impinges on the domain boundary when the bubble is large. I run the simulation for $R_D = 0.6, 0.8, 1$ and 1.5, with $\delta r = 5 \times 10^{-3}$, which is a refinement by a factor of 4 in the radial direction, compared with the previous simulations. With $\delta r = 5 \times 10^{-3}$ the convergence of $\max(R_{int})$ with increasing R_D is third-order, as for the one-dimensional results of Chapter 5. This increase in resolution leads to an increase in computational costs by more than one order of magnitude, and so in general I continue to set $\delta r = 0.02$.

The similarity of the results for one- and two-dimensional simulations shown in Figure 6.1 and Table 6.1 demonstrates that asymmetries in the two-dimensional model are not the limiting factor on the performance of the boundary conditions for these simulations of an air gun bubble.

Due to the stiffness of the equation of state for water, and the high density of water, the relative changes in density of the water are very low. If the simulation is run using single precision floating point variables, the relative changes in density on the boundary when $R_D = 1$ are so low that machine precision errors are introduced. In order to avoid this, the simulations are run with all floating point variables calculated to double precision. This moves the point at which machine precision errors begin to

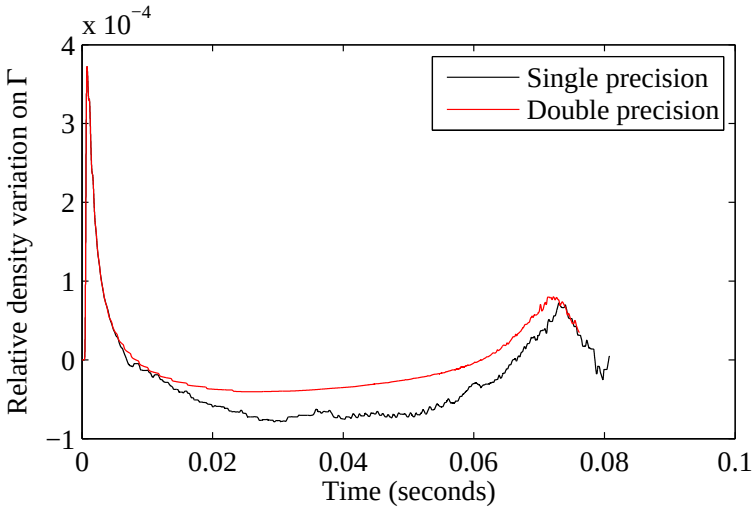


Figure 6.3: Relative variation of density with time at the point on Γ where $\theta = \pi/2$ when $R_D = 1$, for single (black line) and double (red line) precision simulations. The relative variation of density is calculated as $(\rho_t - \rho_0)/\rho_0$, where $\rho_t = \rho(R_D, \pi/2, t)$ and $\rho_0 = \rho(R_D, \pi/2, 0)$.

have an effect further from the bubble, beyond the domain boundary. Figure 6.3 shows the relative changes in density at the domain boundary for simulations using single (black line) and double (red line) precision floating point variables. Note that after the initial pulse the trace for the single precision simulation becomes noisy, as errors due to machine precision accumulate.

6.4 Surface instabilities

A planar interface between two fluids, subject to an acceleration or gravitational field perpendicular to its plane may be stable or unstable, depending on the direction of the acceleration and the relative densities of the two fluids. Various factors affect the extent of the instability. For example, surface tension and viscosity stabilise the smallest disturbances. Although not the first to observe the phenomenon, Taylor (1950) gives a derivation of the stability criteria for this scenario, which uses a linearisation of the equations of motion to calculate the criteria for the stability of an interface

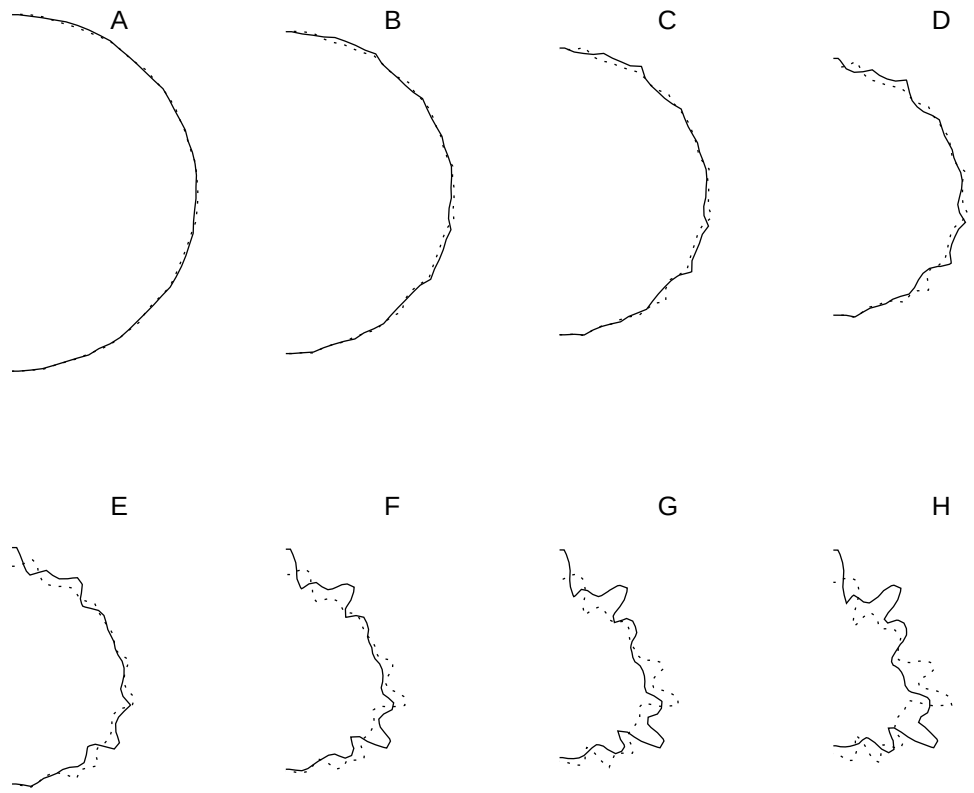


Figure 6.4: The shape of the bubble at different times during collapse, with $R_D = 1$ and $g = 0$. The initial shape of the bubble is a sphere subject to a sinusoidal disturbance η. The solid line corresponds to $\eta = \eta_{\sin} = 0.001 \sin(2n\theta/5)$. The dashed line corresponds to $\eta = \eta_{\cos} = 0.001 \cos(2n\theta/5)$. (A) $t = 29.2$ms; (B) $t = 50.2$ms; (C) $t = 56.6$ms; (D) $t = 61.4$ms; (E) $t = 64.6$ms; (F) $t = 67.0$ms; (G) $t = 68.7$ms; (H) $t = 70.4$ms.

to small disturbances, in an analysis which is now relatively commonplace. Taylor (1950) shows that when the acceleration of the interface between two fluids is directed towards the heavier fluid, the interface is unstable. This form of instability is well known, and is referred to as the Rayleigh-Taylor instability. It has been observed that this form of instability affects oscillating bubbles during the late stages of collapse, when the bubble wall is accelerating outwards (Taylor and Davies, 1943). The assumption that the characteristic length scales of disturbances are small compared with the size

of the bubble allows the bubble surface to be considered planar, and the analysis of Taylor (1950) may be applied. In Appendix C I derive stability criteria for the surface of a bubble, following Taylor (1950). The two-dimensional results should be unstable to all sizes of disturbance. However, due to the coarseness of the mesh, fine disturbances cannot be resolved. There is numerical viscosity inherent in the model, which contributes to the increased stability of the modelled surface compared with theory.

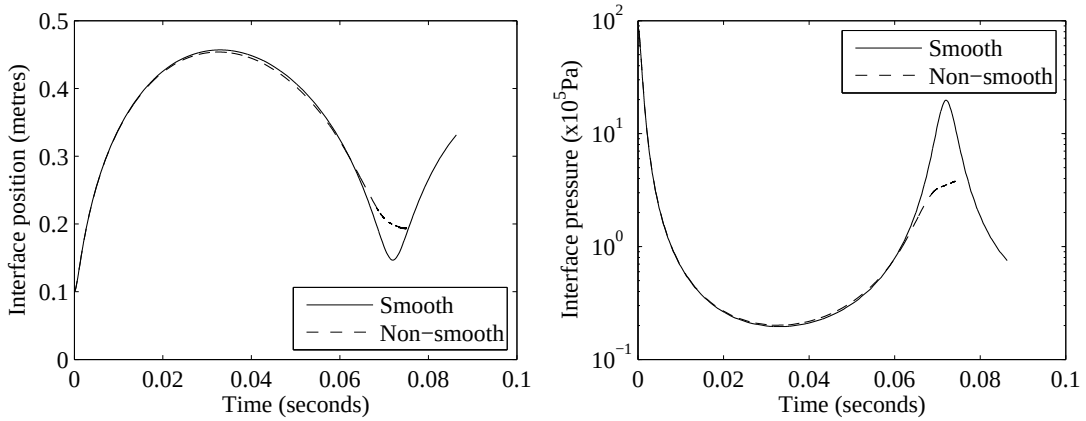


Figure 6.5: Variation of interface position (left) and interface pressure (right) with time for a smooth (solid line) and a non-smooth (dashed line) bubble.

I run the two-dimensional model with $g = 0$, and with the bubble initially subject to a small sinusiodal disturbance, so that the interface position is given by $R = 0.1 + \eta(\theta)$ as in Section 6.2, where η is set either to $\eta_{\sin} = 0.001 \sin(2n\theta/5)$ or $\eta_{\cos} = 0.001 \cos(2n\theta/5)$, where n is the number of cells in the polar direction, as in Chapter 4. Figure 6.4 shows the bubble shape at various times for these two cases. During the expansion stage of oscillation (not shown in Figure 6.4), the disturbances to the bubble shape are stable and the bubble surface remains relatively smooth. As the bubble collapses, the interface becomes unstable. The small disturbances to the bubble shape grow with time, and Rayleigh-Taylor fingers develop at the interface.

Figure 6.4 appears to show the growth of the disturbance before the acceleration of the interface is directed towards the water. There is a competing argument for the instability of the bubble surface, which states that during collapse any asymmetries are

magnified (Cox *et al.* (2004)). This argument implies the bubble should be unstable for the entire period of collapse. This effect can be seen in Figure 6.4, where the disturbances begin to grow when $\dot{R}_{int} < 0$. This observation is in agreement with the images presented by Langhammer and Landrø (1996), where the surface appears to develop wrinkles throughout the period of bubble collapse.

Note that the eventual shape of the bubble is sensitive to the initial disturbances. The slight difference between $\eta_{\sin}$ and $\eta_{\cos}$ leads to significant differences in bubble shape towards the end of the bubble collapse. The Rayleigh-Taylor fingers present in Figure 6.4 are almost as fine as can be supported by the coarse grid. The stability analysis in Appendix C shows that the interface should be unstable to all sizes of disturbances at the end of collapse. I anticipate that grid refinement would simply allow the Rayleigh-Taylor fingers to be resolved in more detail.

During the late stages of bubble collapse ((F) to (H) in Figure 6.4) there is a long Rayleigh-Taylor finger extending along the polar axis. Polar coordinate systems contain a singularity at the poles. It is well known (Kane *et al.* (2000), Kifonidis *et al.* (2003) and Kifonidis *et al.* (2006)) that this singularity causes faster growth rates of Rayleigh-Taylor instabilities at the poles. This phenomenon is a numerical artefact of the discretisation. This is the effect I observe in the final frames of Figure 6.4.

Figure 6.5 shows the variation of R_{int} and P_{int} with time for a smooth bubble and a bubble subject to the initial disturbance $\eta_{\cos}$. Note, that if $\eta = \eta_{\sin}$ Figure 6.5 would be the same, to within 0.3% prior to $t = 0.07$. In both cases, gravity is neglected. The smooth bubble is spherically symmetric. For most of the oscillation the shape of the bubble has negligible effect on R_{int} and P_{int}, as the amplitudes of the disturbances are small. Towards the end of the bubble collapse, as the amplitudes of the disturbances increase, they begin to have an effect on the interface position and interface pressure. The minimum volume of the bubble during collapse is increased. The collapse is slower for the non-smooth bubble, and the peak interface pressure on collapse is reduced. Clearly the growth of the disturbances has a damping effect on the motion of the bubble. Consequently it has a damping effect on the signals emitted by the bubble. Given the

limitations of the numerical scheme imposed by the resolution, the omission of surface tension and viscosity, and the presence of numerical viscosity, it is not possible to ascertain the extent to which the bubble is damped due to the growth of disturbances in reality. Whilst the large scale motion of the bubble is dependent on the degree to which the surface wrinkles, the interchangeability of $\eta_{\sin}$ and $\eta_{\cos}$ in Figure 6.5 shows that R_{int} and P_{int} have very little dependence on the exact shape of the interface.

The focus of this work is not the development of multi-fluid models. However, this work has highlighted a potential damping mechanism for air gun bubbles. Further investigations into these effects will require significantly more effort, and could be better achieved by applying the NLAA boundary condition in existing an finite volume code with an accurate and robust interface model. Such a model might enable the quantification of the extent to which bubble motion is damped due to the growth of disturbances.

6.5 Bubble rise

Air bubbles in water rise due to bouyancy. This effect is visible when gravity is included in the present model. Figure 6.6 shows velocity vectors in and around the bubble at different stages during the oscillation. Figure 6.7 shows contours of velocity magnitude at the same instants in time. In both Figures, frame (A) shows the bubble during rapid expansion. The velocity is directed very close to radially, the contours of velocity magnitude look symmetric and there is little variation with θ. Frame (B) is close to the instant of maximum expansion. The radial velocity at the interface is very low, and the velocity within the bubble is clearly directed upwards. Frame (C) shows the bubble during rapid collapse. The velocity is predominantly in the radial direction, but within the bubble the vertical component of velocity is visible. The asymmetry of the bubble can clearly be seen at this stage in the contour plot, where the velocity magnitudes are higher on the lower half of the bubble, and are roughly centred around a point a small distance vertically above the origin. Note that the fact the contours are

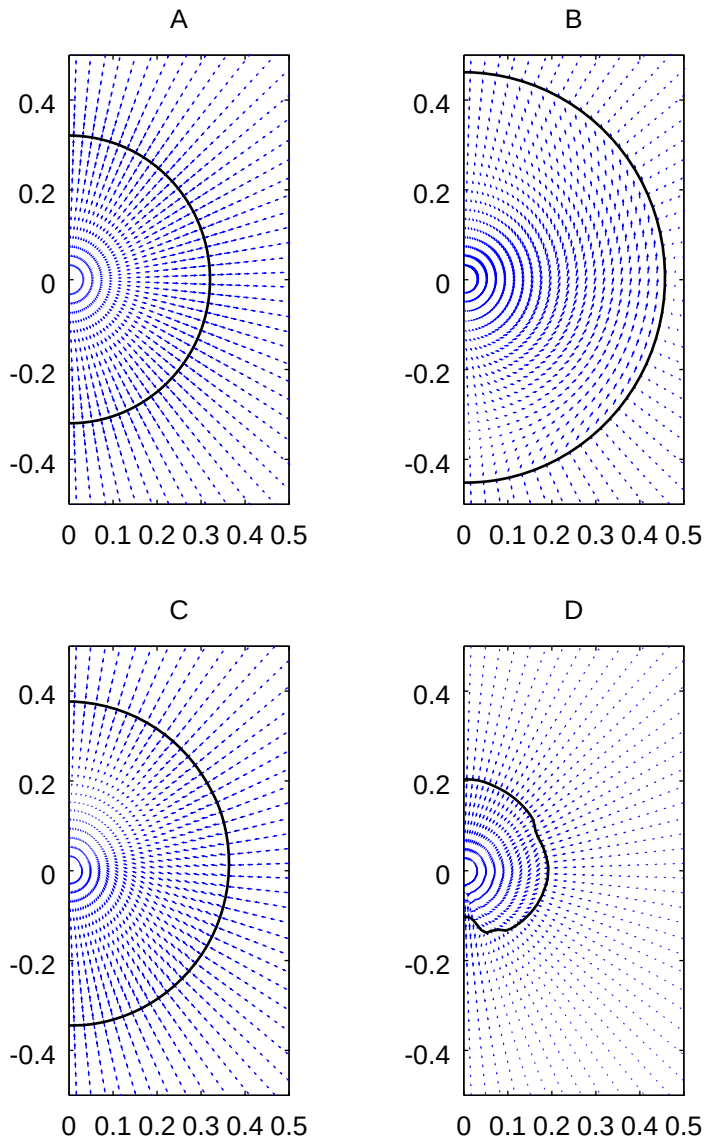


Figure 6.6: Velocity vectors during different stages of bubble oscillation. The abscissa and the ordinate represent the x- and z- coordinates respectively, in metres. The black line traces the shape of the bubble. (A) $t = 9.2$ms during rapid expansion; (B) $t = 33.6$ms at point of maximum expansion; (C) $t = 56.8$ms during early stage of collapse; (D) $t = 70.3$ms in final stages of collapse.

directed radially very close to the origin is an artefact of the polar coordinate system. Frame (D) shows the bubble as it nears the point of maximum collapse. The surface has become unstable, and the contours of velocity magnitude show slight wrinkles. On

the underside of the bubble a jet is forming, piercing upwards into the bubble. This jet is discussed further in the next sections.

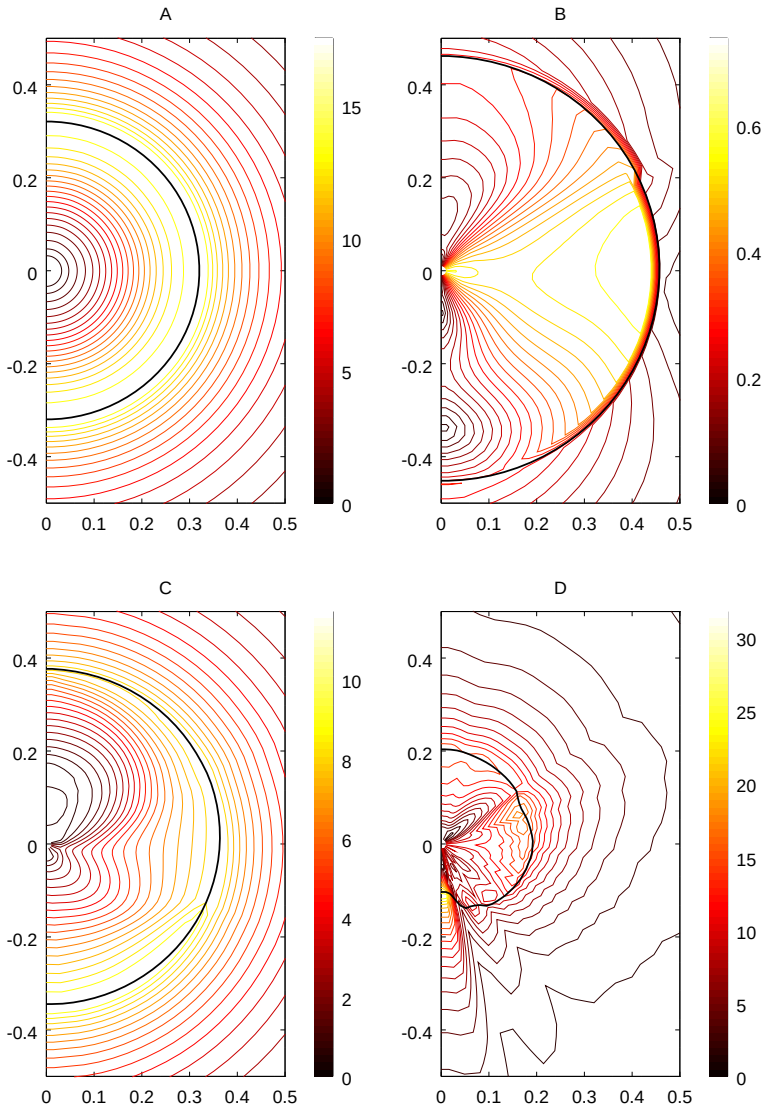


Figure 6.7: Contours of velocity magnitude at different stages of bubble oscillation. The abscissa and the ordinate represent the x- and z- coordinates respectively, in metres. The units of the colour scale are metres per second. The black line traces the shape of the bubble. (A) $t = 9.2$ms during rapid expansion; (B) $t = 33.6$ms at point of maximum expansion; (C) $t = 56.8$ms during early stage of collapse; (D) $t = 70.3$ms in final stages of collapse.

It is well known that oscillating bubbles do not rise with constant velocity. Herring

(1941) and Taylor (1942) first put forward the idea that an oscillating bubble will rise slowly when it is large, and fast when it is small. Ziolkowski (1970), Cox *et al.* (2004) and Barras *et al.* (2012) have all described this behaviour. It has been observed that air gun bubbles typically rise at approximately one metre per second (Ziolkowski and Johnston, 1997). The rise rate of air gun bubbles has been captured in previous models by Cox *et al.* (2004).

Herring (1941) and Taylor (1942) derive and investigate an equation for the vertical motion of a bubble due to gravity. The equation is

$$-\frac{dz_b}{dt} = \frac{2g}{R(t)^3} \int_0^t R(\tau)^3 \, d\tau, \tag{6.2}$$

where z_b is the vertical position of the centre of a spherical bubble, $R(t)$ is the time-dependent radius of the bubble and g is the acceleration due to gravity. This equation has been used in air gun modelling in Ziolkowski (1998).

There are clearly problems with this equation. Most notably, there is no dependence on the relative densities of the bubble and the fluid. According to this model a bubble with a density of 1kgm^{-3} would rise at the same rate as a bubble with density 100kgm^{-3}. The derivation of this equation can be outlined as follows: (1) calculate the momentum in the water around a bubble translating at a certain velocity; (2) relate the rate of change of momentum in the water to the rate at which momentum is transmitted from the bubble to the water and (3) integrate. The problem with this derivation is that the approximation of the motion of the water around the translating bubble is a very simplistic one, in which there is no turbulent motion or jet formation. When this equation is used with the results of the one-dimensional model in Chapter 5, the rise rate of the bubble at the point of collapse peaks at 50ms^{-1} every oscillation, and in the longer term, the bubble is found to rise approximately 20 metres every second. Cox *et al.* (2004) show similarly erroneous results when gravity is included in what is effectively the same manner, but without the explicit use of equation 6.2. This is clearly incorrect, and in error by a factor of about 20. It is also worth noting that if the bubble were to translate at 50ms^{-1}, it would undergo very significant deformation, and surface instabilities would probably cause the bubble to disintegrate into many smaller

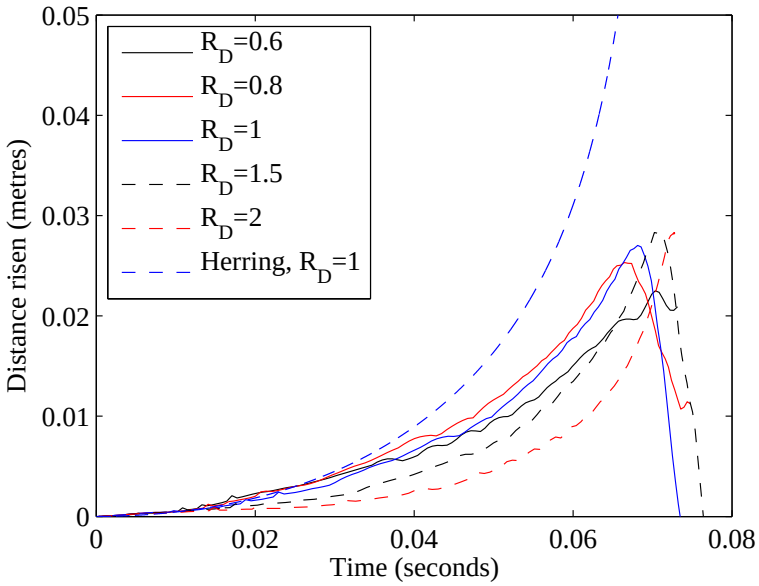


Figure 6.8: Variation of the vertical location of the bubble with time, for the two-dimensional simulation with $R_D = 0.6, 0.8, 1, 1.5, 2$, and using the formula of Herring (1941) on data from the two-dimensional model with $R_D = 1$.

bubbles. The derivation of Herring (1941) fails to account for all the momentum in a body of water around a translating oscillating bubble, and hence gives an erroneously low added mass, causing an over-estimate of the translation velocity.

For the two-dimensional results I numerically integrate the density over the bubble to obtain the vertical position of the centre of mass, according to

$$z_b = \frac{1}{\int\limits_{V_b} \rho dV_b} \int\limits_{V_b} \rho \mathbf{r} \cdot \mathbf{e_z} dV_b, \tag{6.3}$$

where V_b is the volume of the bubble. z_b is the z-component of the first moment of mass of the bubble. Figure 6.8 shows a plot of the variation of vertical location of the bubble with time for the two-dimensional model, for a range of values of R_D. Figure 6.8 also shows the vertical location of the bubble as calculated using the equation of Herring (1941) on the results of the two-dimensional model with $R_D = 1$. The results using Herring's equation are clearly in disagreement with the results of the two-dimensional model, although they agree during the very early stages of expansion. The results of the two-dimensional model roughly agree with those of Cox *et al.* (2004), giving

approximately 2.5cm of rise in the first oscillation. However, the rise rate obtained from the two-dimensional model varies significantly with R_D, although there is no uniform trend over the range of R_D modelled. Also note that in some cases (most obviously $R_D = 1$) the bubble descends during the final stages of collapse. This is an error, and it may be associated with the polar coordinate system, as the bubble gets 'stuck' trying to pass the origin. The results clearly show a sensitivity of the rise rate, and the lack of a uniform trend with R_D suggests there are several factors affecting the vertical motion of the bubble. A more robust model, capable of simulating multiple bubble oscillatings, would be of benefit for investigations into bubble rise rates. Such a model would be best constructed in cylindrical coordinates, to avoid the singularity at the origin.

6.6 The formation of jets

When an air bubble in water is oscillating in the presence of a gravitational field, the background pressure field is non-uniform due to hydrostatic pressure. Any asymmetries in the bubble are amplified as it collapses. During collapse, the high pressure on the underside of the bubble causes it to collapse faster here than elsewhere, leading to the formation of a jet of water which travels upwards and pierces through the bubble. This has been observed in a variety of types of bubbles, including underwater explosions and air gun bubbles. Similar jets can form when air bubbles in water are impacted by shock waves, or collapse close to a solid boundary. These jets have been experimentally observed by, for example, Tomita and Shima (1986) and Bourne and Field (1992) and modelled using finite volume methods (Miller *et al.* (2013) and Hu and Khoo (2004)), boundary integral methods (Cox *et al.*, 2004), and smoothed particle hydrodynamics (Colagrossi and Landrini (2003) and Zhang *et al.* (2013a)).

Figure 6.9 shows the shape of the bubble at different times during collapse, both with (solid line) and without (dashed line) gravity. During the early part of collapse (frames (A) to (E)) the bubble remains roughly spherical, and translates upwards without much

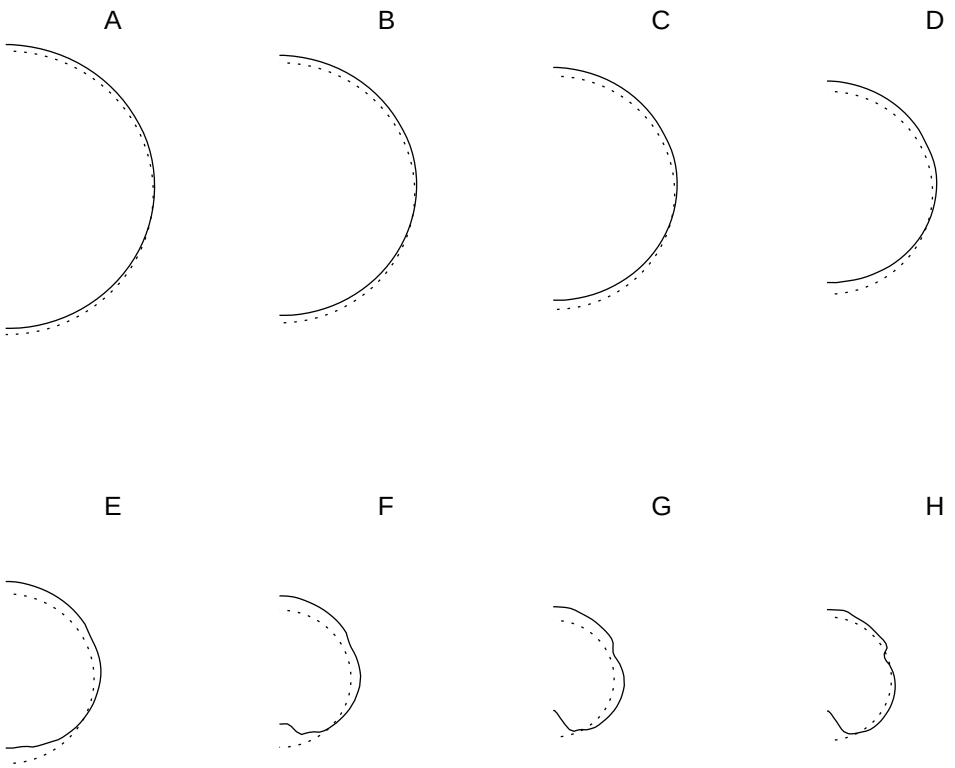


Figure 6.9: The shape of the bubble at different times during collapse, both with (solid line) and without (dashed line) gravity. (A) $t = 56.8$ms; (B) $t = 59.8$ms; (C) $t = 62.7$ms; (D) $t = 65.3$ms; (E) $t = 67.7$ms; (F) $t = 69.9$ms; (G) $t = 71.8$ms; (H) $t = 73.6$ms.

deformation. Towards the end of collapse (frames (F) to (H)) the bubble deforms, and a jet on the underside develops, although the simulation breaks down before this jet pierces the bubble. Note also that the surface is unstable during the late stages of collapse, and slight perturbations to the shape of the upper half of the bubble are also visible in frames (F) to (H).

As discussed in Section 6.4, there is a singularity at the poles which increases the speed of growth of instabilities. In long run time simulations, these errors can evolve into axial jets (Kifonidis *et al.*, 2006). As the physical jet directed radially inwards

approaches the origin, the unphysical jet forms in the opposite direction. The collision of these jets eventually leads to the breakdown of the simulation. If the simulation could continue without this error, I would expect to observe the jet piercing right through the bubble, which would become toroidal in shape, as in Cox *et al.* (2004). Obviously, for a real air gun bubble, the physical presence of the gun prevents this. During operation the system of air guns is not stationary, but is towed through the water, making the situation even more complex. It should be noted that if the code did not break down due to these instabilities, the passage of the interface across the singularity at the origin would probably cause problems. Whilst the singularity along the polar axis is also present in cylindrical coordinate systems, it is weaker, and the singularity at the origin is removed. The NLAA boundary condition is designed for polar coordinate systems, and an implementation of the NLAA boundary condition in a version of this model based in cylindrical coordinates was found to be unstable, as discussed in Chapter 4, Section 4.9. An improvement to the present scheme would be an irregular grid in a spherical domain. If the grid were cylindrical near the origin, and varying to polar on Γ, some of the problems observed in the present scheme could be avoided.

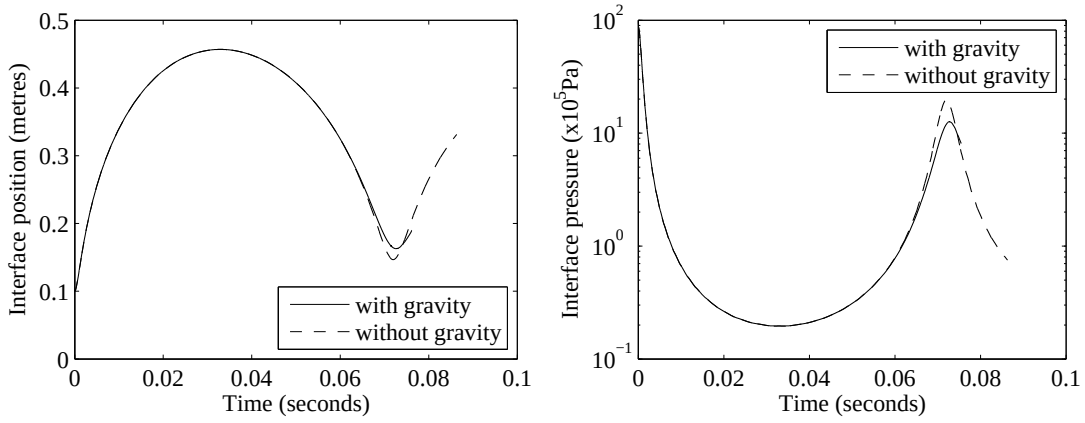


Figure 6.10: Variation of interface position (left) and interface pressure (right) with time with (solid line) and without (dashed line) gravity.

Figure 6.10 shows the time variation of the interface position and interface pressure for the bubble with (solid line) and without (dashed line) gravity. There are small differences in both interface position and interface pressure during the first half of the

oscillation, but not more than 0.5%. It is only during the late stages of bubble collapse that the differences between the two cases become significant. From Figure 6.10 it appears that a consequence of the translation and deformation of the bubble due to gravity is to damp the motion, such that the bubble does not collapse to such a small volume, and the peak pressure on collapse is reduced. This effect can be explained by considering that the translation of the bubble (and the water surrounding it), the deformation of the bubble and the jet on the underside of the bubble, all require energy. The energy associated with this deformation and translation is not available for oscillation, and so the oscillatory motion is more damped.

6.7 Conclusions

I present detailed results of two-dimensional air gun bubble simulations. I set out to do this for two reasons. Firstly, to show the boundary condition works well, and secondly, to highlight some of the phenomena present in air gun bubbles. The results demonstrate that my boundary condition is not limited to one dimension. It enables long run time simulations of two-dimensional air gun bubbles on highly truncated domains. Results show a second-order convergence as the boundary is moved towards the far field if a coarse grid is used, although the contribution of errors due to the boundary condition is third-order. The results of simulations with a refined grid demonstrate this third-order convergence. The model captures various asymmetric bubble phenomena. The model captures the instability of the bubble surface, and it can be seen that the instability occurs at all times when the bubble is contracting, contrary to the standard Taylor (1950) stability criteria. The rise of the bubble due to gravity, the deformation of the bubble as it rises, and the formation of a jet as the bubble collapses are also captured by the model. These phenomena are shown to have a damping effect on bubble motion. The boundary condition allows simulations which can capture these phenomena to be carried out on highly truncated domains, reducing computational costs. The reduction in costs facilitates research into these interesting phenomena. This is the main value of the NLAA boundary condition.

Chapter 7

Viscosity

7.1 Introduction

In the previous chapters I neglect viscosity, and consider only the Euler equations for inviscid flow. In their investigations into underwater explosions Taylor and Davies (1943) observed that the energy loss due to viscous forces is very small compared with the whole energy of the bubble. This argument justified the assumption of inviscid flow in their calculations. By neglecting viscosity, they were able to dimensionally match small scale laboratory experiments with the underwater explosions of interest. Air gun bubbles were first modelled by Ziolkowski (1970), who also argued that viscosity can be neglected. In an effort to introduce damping into modelled bubble motion, various authors (Dragoset (1984), Laws *et al.* (1990) and Landrø (1992)) introduced viscous terms into air gun bubble models, based on the approach of Bornhorst and Hatsopoulos (1967). Langhammer and Landrø (1993a) conducted experiments in which a small air gun was fired in a tank, and the viscosity of the water was varied. They demonstrated that the observed changes in air gun signatures could not be accounted for by viscous terms of the form of Bornhorst and Hatsopoulos (1967). Langhammer and Landrø (1993a) suggested that with increased viscosity, less energy is transferred from the

bubble to the water (in the form of turbulent motion) and hence the bubble retains more energy for oscillation.

In this chapter I alter my model to include viscosity, and hence solve the Navier-Stokes equations. Viscosity has a significant effect on the bubble shape, due to the sensitivity of the bubble shape to initial conditions. Viscosity appears to have only a minor effect on the bubble signatures, and this effect is in agreement with the observations of Langhammer and Landrø (1993a). Results also show a significant effect of viscosity on the bubble rise rate, which is shown to be due to viscosity at the bubble surface.

The layout of this chapter is as follows. In Section 7.2 I give details of the full Navier-Stokes equations, including viscous terms. In Section 7.3 I describe the implementation of the viscous terms in the numerical scheme. In Section 7.4 I present simple order of magnitude arguments to estimate the likely scales of turbulent motion present. In Section 7.5 I present the results of the model with and without viscosity, and discuss the effects of viscosity on the shape of the bubble and the signals emitted by the bubble. Section 7.6 is a summary of conclusions.

7.2 Navier-Stokes equations

The compressible Navier-Stokes equations may be expressed by adding terms for the viscous forces to the Euler equations (equation 4.1). In polar coordinates subect to polar axisymmetry, and in the absence of heat conduction, the compressible Navier-Stokes equations can be written

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial r} + \frac{\partial \mathbf{G}(\mathbf{U})}{\partial \theta} + \mathbf{S_r}(\mathbf{U}) + \mathbf{S}_\theta(\mathbf{U}) = \mathbf{D}(\mathbf{U}) + \mathbf{F}_\mu, \tag{7.1}$$

where, $\mathbf{U}$, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{S_r}$, $\mathbf{S}_\theta$ and $\mathbf{D}$ are as defined in Chapter 4, Section 4.2. The viscous terms $\mathbf{F}_\mu$ are defined by

$$\mathbf{F}_\mu = (0,\, \nabla \cdot \tau_\mu,\, \nabla \cdot (\mathbf{u}\tau_\mu))^T. \tag{7.2}$$

The velocity vector $\mathbf{u}$ is given by $\mathbf{u} = \begin{bmatrix} u, & v \end{bmatrix}^T$, where u and v are the radial and polar components of velocity respectively. In two dimensional polar coordinates, the

viscous stress tensor τ_μ is

$$\tau_\mu = \begin{bmatrix} \tau_{rr} & \tau_{r\theta} \\ \tau_{r\theta} & \tau_{\theta\theta} \end{bmatrix}, \tag{7.3}$$

and the components of τ_μ are

$$\tau_{rr} = \mu \left(2\frac{\partial u}{\partial r} - \frac{2}{3}\nabla \cdot \mathbf{u} \right), \tag{7.4}$$

$$\tau_{r\theta} = \mu \left(r\frac{\partial}{\partial r}\left(\frac{v}{r}\right) + \frac{1}{r}\frac{\partial u}{\partial \theta} \right), \tag{7.5}$$

$$\tau_{\theta\theta} = \mu \left(2\left(\frac{1}{r}\frac{\partial v}{\partial \theta} + \frac{u}{r} \right) - \frac{2}{3}\nabla \cdot \mathbf{u} \right), \tag{7.6}$$

where μ is the dynamic viscosity. I note that

$$\nabla \cdot \tau_\mu = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \begin{bmatrix} \tau_{rr} \\ \tau_{r\theta} \end{bmatrix} \right) + \frac{1}{r\sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta \begin{bmatrix} \tau_{r\theta} \\ \tau_{\theta\theta} \end{bmatrix} \right), \tag{7.7}$$

due to the polar coordinate system, and that $\mathbf{u}\tau_\mu = (u\tau_{rr} + v\tau_{r\theta},\, u\tau_{r\theta} + v\tau_{\theta\theta})^T$.

7.3 Numerical scheme

I solve equations 7.1 using the numerical scheme described in Chapter 4, with an additional operator splitting step to account for the viscous terms. To recapitulate, the scheme is a first-order Godunov-type scheme, using HLLC fluxes, in combination with a ghost fluid method to capture the fluxes through the interface, which is tracked using a level set. Geometric, gravitational and viscous source terms are accounted for using a first-order operator splitting method, after the solution as been evolved with the single phase Euler solver, before the reconstruction of the two-fluid domain using the ghost fluid method. The spatial derivatives in the viscous terms are calculated using second-order central differences within the domain, and first-order one-sided differences on the boundaries.

The ghost fluid method was developed for use in solving the Euler equations for inviscid, incompressible flow. However, an extension to viscous flows is straightforward. Kang

et al. (2000) gives a version of the ghost fluid method for the incompressible Navier-Stokes equations for viscous flow. A version of the ghost fluid method for viscous compressible flow is presented in Fedkiw and Liu (2001). Terashima and Tryggvason (2009) provide a thorough description of a method of solution of the compressible Navier-Stokes equations using a ghost fluid method, with the inclusion of surface tension. The method in Terashima and Tryggvason (2009) uses a marker-point-based front-tracking scheme instead of a level set to track the interface; however, the extension to viscous flows is unaffected by this choice.

If a fluid is inviscid, and is flowing past a boundary, be that solid, or the interface with another fluid, the component of velocity transverse to the boundary may be discontinuous at the boundary. In other words, there is no boundary layer within the fluid, and at the boundary the fluids can slip. For viscous fluids, a *no-slip condition* must hold at any boundary, meaning that the velocity of the fluid on the boundary must equal the velocity of the boundary. This constrains the velocities transverse to the interface to be continuous across the interface. Recall the ghost fluid method. For inviscid fluids, the velocity component tranverse to the interface is extrapolated across the interface into ghost cells, or taken from the star-states of the interfacial Riemann problem. To simulate viscous fluids, the velocity components transverse to the interface are simply copied into ghost cells directly from the corresponding real cells. This constrains the transverse velocities to be continuous across the interface. All other aspects of the ghost fluid method are unchanged by the inclusion of viscosity. I neglect surface tension.

The viscosity of both air and water depends on temperature. For air I use the empirical-theoretical relation of Sutherland (1893), which is given by

$$\mu_{\text{air}} = \mu_0 \frac{T_0 + C}{T + C} \left(\frac{T}{T_0} \right)^{\frac{3}{2}}, \tag{7.8}$$

where $\mu_0 = 1.716 \times 10^{-5}\text{kgm}^{-1}\text{s}^{-1}$, $T_0 = 273.15\text{K}$ and $C = 110.4\text{K}$. The temperature in the air is calculated from the ideal gas equation of state $T = p/(\rho R)$, where $R = 287\text{Jkg}^{-1}\text{K}^{-1}$ is the gas constant for air. For the International Standard Atmosphere conditions at sea level, with $T = 288.15\text{K}$, Sutherland's equation gives

$\mu_{air} = 1.789 \times 10^{-5}\text{kgm}^{-1}\text{s}^{-1}$. Sutherland's equation is known to be accurate to within 10% for temperatures below 555 Kelvin and pressures below 3.45×10^6Pa. Whilst the temperature of the air in an air gun remains within this range, the pressure does not. For approximately the first 1ms of oscillation, the pressure in the water is greater than 3.45×10^6Pa. Subsequently, including when the bubble collapses, the pressure remains below this limit. During the very early stages of expansion, the motion is very close to spherically symmetric, and the bubble rise velocity is still small. This first part of expansion is probably the stage at which viscosity has least influence on the bubble motion. Hence, it seems reasonable to use the Sutherland equation even though the pressures are not entirely within the limits for which the relation is valid. In the water the temperature is approximately constant, and I use $\mu_{\text{water}} = 10^{-3}\text{kgm}^{-1}\text{s}^{-1}$ (Langhammer and Landrø, 1993a).

7.4 Order of magnitude arguments

The likely extent of turbulent motion within and around the bubble can be estimated using order of magnitude arguments. Whilst these are generally applied to steady-state flows, they can still be used to provide some information about the degree of turbulent motion in transient problems such as this. Useful introductions to turbulence can be found in Tritton (1988) and Frisch (1995), both of which contain much of the analysis which follows. The Reynolds number is a non-dimensional group, the value of which describes the relative importance of inertial and viscous forces in a fluid. The Reynolds number is defined as $Re = \rho u D/\mu$, where ρ is the density, u is a characteristic velocity of the motion, D is a characteristic length scale of the motion and μ is the dynamic viscosity. A low Reynolds number implies that viscous forces dominate the motion, and in a high Reynolds number flow, the inertial forces dominate. Low Reynolds number flows are often laminar, whilst turbulence occurs at higher Reynolds numbers. For incompressible flow, for a given geometry, the Reynolds number is the only control parameter of the flow. This is not the case for compressible flow, such as that within the

bubble, but notwithstanding this, the Reynolds number can provide useful information about the type of motion likely to occur.

The largest scales of turbulent motion are likely to correspond to the scales of the bubble oscillation. The bubble period and the maximum bubble volume are good proxies for a turbulent timescale of $t_{turb} = 0.07$ seconds, and a turbulent lengthscale of $L_{turb} = 0.45$ metres (These values are taken from the results in Chapter 6.). The maximum magnitude of the velocity around the bubble is about 40ms^{-1}, though as a maximum, this is not a good choice for typical characteristic velocity of the turbulence. A good estimate of the velocity can be obtained from L_{turb} and t_{turb} as $u_{turb} = L_{turb}/t_{turb} \approx 6.43\text{ms}^{-1}$. Water has a density of $\rho_w = 1000\text{kgm}^{-3}$, and a viscosity of $10^{-3}\text{kgm}^{-1}\text{s}^{-1}$ (Langhammer and Landrø, 1993a). The average density of the bubble over one period of oscillation is $\rho_a = 5.63\text{kgm}^{-3}$ and the air viscosity is approximately $10^{-5}\text{kgm}^{-1}\text{s}^{-1}$ (Sutherland, 1893). The turbulent Reynolds number is now calculated according to $Re_t = \rho u_{turb} L_{turb}/\mu$. For air, $Re_{t,air} = 1.63 \times 10^6$. For water, $Re_{t,water} = 2.89 \times 10^6$. The Kolmogorov length scale is an estimate of the smallest scales of turbulent motion, and can be calculated according to $\eta_K = L_{turb} Re_t^{-3/4}$. Using the turbulent Reynolds numbers obtained above, The Kolmogorov length scales are $\eta_{K,air} = 9.86 \times 10^{-6}$ metres and $\eta_{K,water} = 6.42 \times 10^{-6}$ metres.

The transition from laminar to turbulent motion occurs in most steady incompressible flows at Reynolds numbers below 10^5 (Tritton, 1988, p. 278-294). Whilst the oscillatory motion and short timescales of an air gun bubble are likely to prevent the developement of fully isotropic turbulence, there will be a degree of turbulence present in and around the bubble, given the Reynolds numbers of the flow. The photographs presented by Langhammer and Landrø (1996) show very uneven bubble surfaces, which will certainly be a source of irregularity in the flow close to the bubble. The Kolmogorov length scales of the motion are almost 4 orders of magnitude smaller than the computational mesh. Turbulence is inherently three-dimensional, and so the present scheme is unable to fully capture turbulent motion. The coarse mesh and the inherent numerical viscosity in the finite volume scheme both contribute to increasing the effective viscosity of the

simulation. This reduces the Reynolds number of the simulation, reducing the extent to which any turbulent motion can be captured.

The rate at which kinetic energy is dissipated through turbulent motion per unit mass ε can be estimated according to $\varepsilon \approx u_{turb}^3/L_{turb}$. However, the degree of turbulent motion is clearly neither uniform in space nor time. For incompressible flow, the velocity field varies with $1/r^2$ (Rayleigh, 1917), and so it is perhaps reasonable to assume that u_{turb} and L_{turb} also vary inversely with r^2. In this case, both the Reynolds number and the dissipation rate can be shown to vary with $1/r^4$. In theory, the quantity $\int_{R(t)}^{\infty} \rho\varepsilon(r)\, 4\pi r^2 dr$, where $R(t)$ is the bubble radius at time t, could be calculated to determine the rate at which energy is dissipated (Perhaps a better outer limit would be r_{100} where $Re(r_{100}) = 100.$). A comparison between this quantity and the rate at which energy is transmitted into the water by the bubble would allow an estimate of the damping due to viscous terms to be obtained. Both Re and ε have a sixth-order dependence on L_{turb} at the bubble wall, and so the result would be severely affected by the choice of L_{turb}. However, if measurements of the turbulence around the bubble were used to constrain this type of analysis, a useful estimate of the importance of turbulence could be calculated.

7.5 Numerical Results

The initial conditions are similar to those in Chapter 6, and are given by

$$(\rho,\, u,\, p,\, \gamma,\, p_c) = \begin{cases} \big(102,\, 0,\, 8.85 \times 10^6 + p_h,\, 1.4,\, 0\big) & \text{if } 0 \le r \le 0.1, \\ \big(1000,\, 0,\, p_w,\, 7.0,\, 3 \times 10^8\big) & \text{if } 0.1 < r \le R_D, \end{cases} \tag{7.9}$$

where $p_w = p_{atm} + 1000g \times 7.7 + p_h$, $p_{atm} = 1.01325 \times 10^5$, $p_h = -\rho g r \cos\theta$, and $g = 9.81\text{ms}^{-2}$. The units of density, velocity and pressure are kgm^{-3}, ms^{-1} and Pa, respectively. In all cases, I set $R_D = 1$, $\delta\theta = \pi/50$ and use a CFL number of 0.8. Unless otherwise stated, $\delta r = 0.02$. Inviscid simulations are acheived by setting $\mu = 0$ everywhere, and using the inviscid GFM, as described in Chapter 4.

7.5.1 Bubble structure

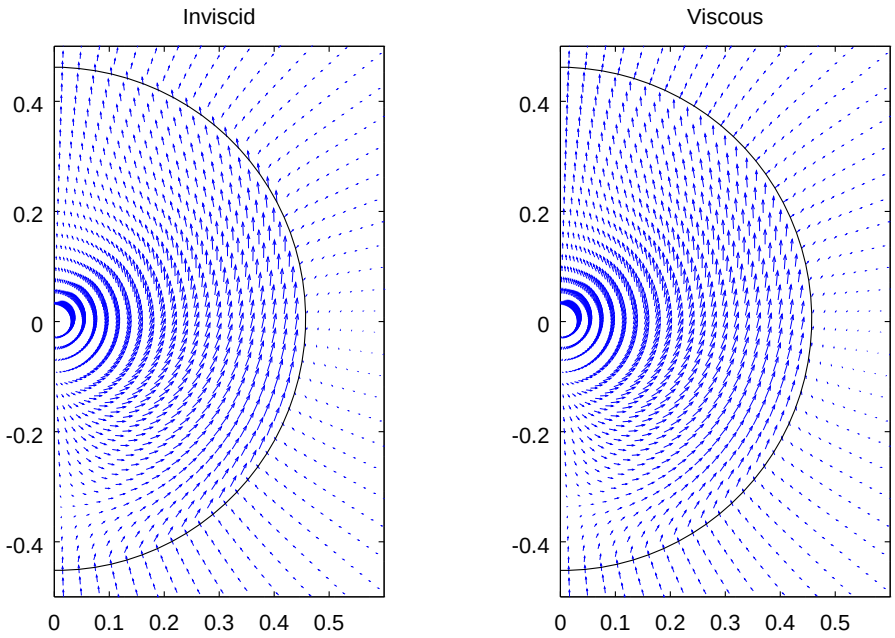


Figure 7.1: Velocity vectors around the bubble at time $t = 33.6$ms for inviscid (left) and viscous (right) models. The abscissa and the ordinate represent the x- and z-coordinates respectively, in metres. The solid black line is the air-water interface.

Figure 7.1 shows the velocity vectors around an air gun bubble, both with (right) and without (left) the viscous terms included, at $t = 33.6$ms. Figure 7.2 shows contours of velocity magnitude around the bubble at the same time, again both with (right) and without (left) the viscous terms included. This instant in time corresponds approximately to the point of maximum expansion. In both cases, there is a change in both the direction and magnitude of the velocity at the interface. Viscosity causes a reduction in the magnitude of the velocity disconuity, as is most clearly visible in Figure 7.2. The countours of velocity magnitude show a change close to the interface in the viscous case which is not present in the inviscid case. It can be seen in Figure 7.1 (by looking closely at a point roughly $(x, z) = (0.45, 0)$) that a small amount of upward momentum has been lost by the air in the bubble through the no-slip constraint at the air-water interface. Note that in the inviscid case, the maximum velocity in the bubble

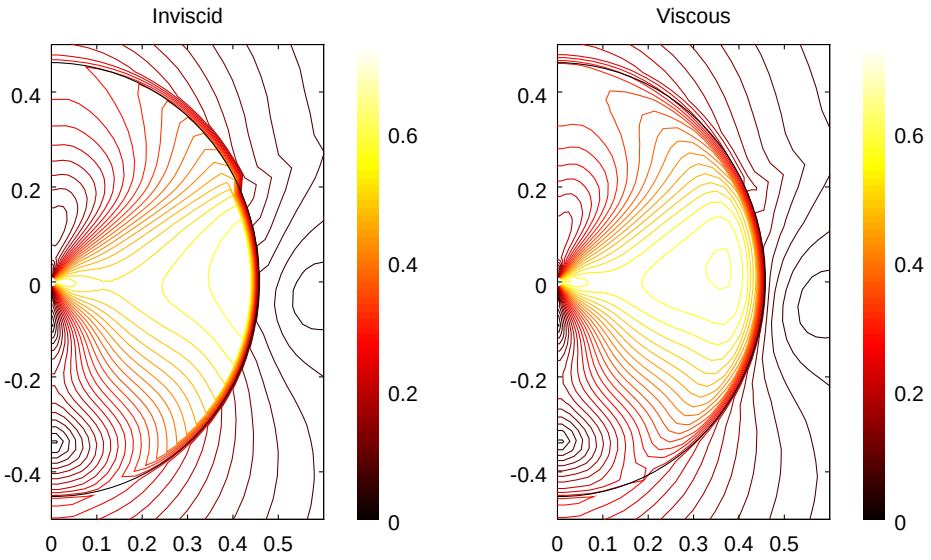


Figure 7.2: Contours of velocity magnitude at $t = 33.6$ms for inviscid (left) and viscous (right) models. The abscissa and the ordinate represent the x- and z-coordinates respectively, in metres. The solid black line is the air-water interface.

occurs at the interface, whilst in the viscous case it occurs approximately 0.1 metres away from the interface.

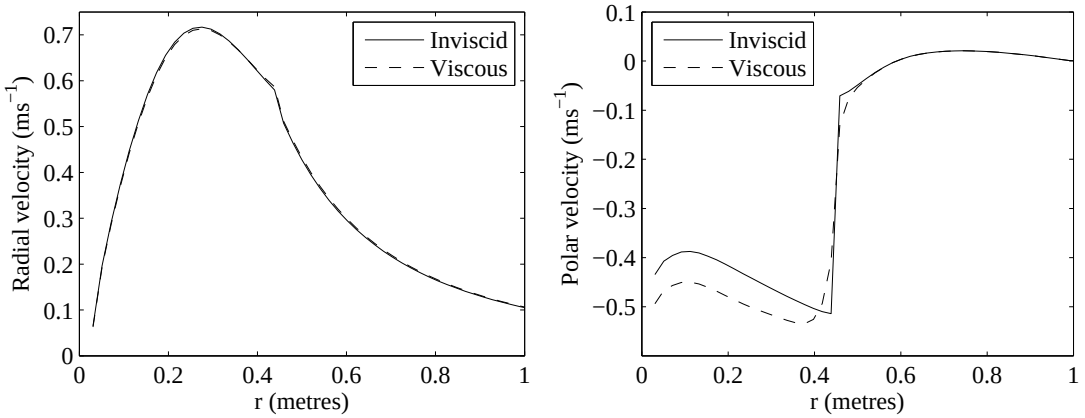


Figure 7.3: Variation of radial and polar velocity components with radius along $\theta = \pi/2$ at $t = 32.2$ms, for inviscid (solid line) and viscous (dashed line) models. The interface is located at approximately $r = 0.45$.

Figure 7.3 shows the radial and polar components of velocity along the line $\theta = \pi/2$

just before the bubble reaches its maximum volume, at $t = 32.3$ms. The solid line corresponds to the inviscid model, and the dashed line to the viscous model. The location of the interface is approximately $r = 0.45$. Note that in both cases the radial velocity component is continuous, although with discontinuous gradient, across the interface. For the inviscid model, the polar component of velocity is discontinuous. In the viscous model, the polar velocity is continuous across the interface. There is still a large change in the polar component of velocity across the interface, but it varies smoothly. Far from the bubble, the velocity fields are very similar for the inviscid and viscous models, which re-inforces the justifications of Taylor and Davies (1943) and Ziolkowski (1970) and the results of Langhammer and Landrø (1993a) that viscosity has little effect on the signals emitted by the bubble, once they are propagating through the water. Within the bubble there are significant discrepancies between the viscous and inviscid models, as the no-slip condition causes a boundary layer within the bubble, and the effects of this small change accumulate over time leading to larger differences in velocity fields. Figure 7.4 shows contours of the magnitude of polar velocity for both inviscid (left) and viscous (right) models, at the same instant in time, $t = 32.2$ms. At this point in time the bubble is still very close to spherical, and has risen only a very small distance. Therefore, the polar velocity component is a close proxy for the transverse velocity at the interface. The boundary layers on either side of the interface are clearly visible in Figure 7.4. The resolution of the boundary layers either side of the interface is limited by the computational mesh ($\delta r = 0.02$), though it is clear that a boundary layer with appriximate thickness 0.06 metres exists in the bubble, and in with approximate thickness 0.03 metres the water. Figure 7.5 shows contours of the polar component of velocity at $t = 32.5$ms (roughly the same point during oscillation as in Figure 7.4), but with $\delta r = 0.01$, an increase by a factor of two in the mesh resolution in the radial direction. Whilst the increase in resolution has led to changes in the bubble structure, the boundary layer thicknesses are unchanged.

Based on the polar velocity component at the interface, the maximum polar velocity in the bubble, and the fact that far from the bubble the polar velocity falls to zero, I estimate free-stream velocities for the boundary layers to be $V_{fs,air} = 0.4\text{ms}^{-1}$ and

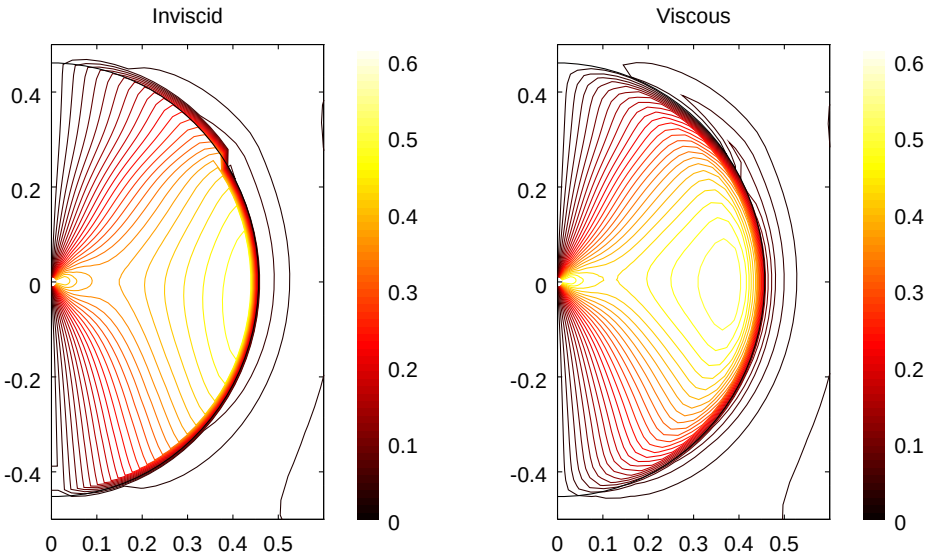


Figure 7.4: Contours of the magnitude of the polar component of velocity at $t = 32.2$ms for inviscid (left) and viscous (right) models, with $\delta r = 0.02$. The abscissa and the ordinate represent the x- and z-coordinates respectively, in metres. The solid black line is the air-water interface.

$V_{fs,water} = 0.2\text{ms}^{-1}$. Using the viscosities and length scales set out in Section 7.4, and a density of air at the instant of maximum expansion of 1.1kgm^{-3}, I obtain Reynolds numbers for the asymmetric motion within the bubble of $Re_{air} = 2.0 \times 10^4$ and outside the bubble of $Re_{water} = 9 \times 10^4$. This difference in Reynolds numbers shows that the relative importance of vicous forces in the water is less than in the air. This contributes to the small influence of viscosity on the motion away from the bubble compared with the influence of viscosity within the bubble. Boundary layers are thicker in regions with lower Reynolds numbers, which is clearly the case in Figures 7.3, 7.4 and 7.5.

7.5.2 Bubble shape

Figure 7.6 shows the shape of the bubble at various stages of collapse, both with (dashed line) and without (solid line) viscosity. The shapes closely match during the early stages of collapse (frames (A) to (C)), but begin to differ towards the end of collapse (frames

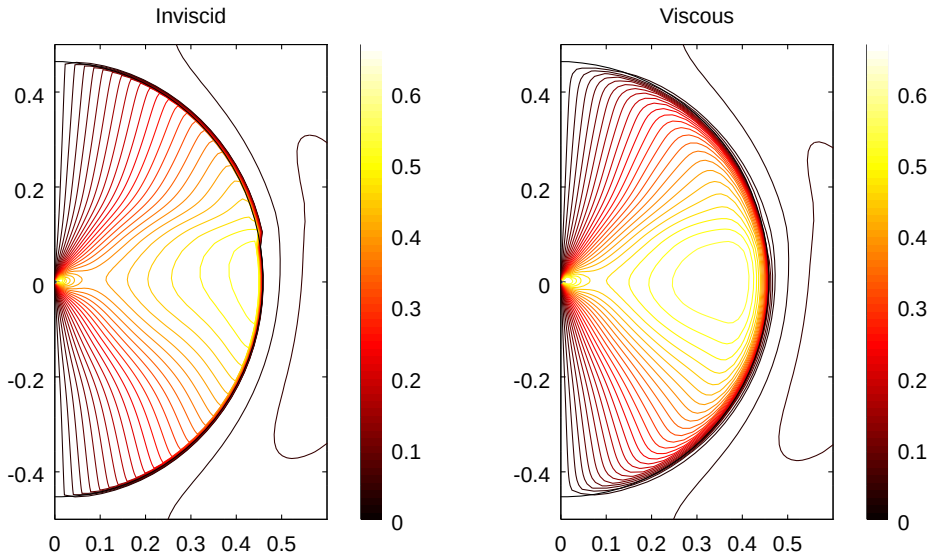


Figure 7.5: Contours of the magnitude of the polar component of velocity at $t = 32.5$ms for inviscid (left) and viscous (right) models, with $\delta r = 0.01$. The abscissa and the ordinate represent the x- and z-coordinates respectively, in metres. The solid black line is the air-water interface.

(D) to (F)). This is unsurprising. At this stage in the oscillation, the surface of the bubble is unstable, and the bubble shape at the end of collapse is highly sensitive to small changes in shape during prior to collapse. This sensitivity is also observed in Chapter 6, Section 6.4. Figure 7.6 also shows a small reduction in rise rate when viscosity is included. This is visible most clearly in frames (C) to (E), and is discussed in more detail later.

Figure 7.6 shows that the shape of the bubble during collapse is different for the viscous and inviscid models. Figure 7.7 shows the shape of the bubble at two times during collapse. Frame (A) is at $t = 69.9$ms, and frame (B) is at $t = 73.7$ms, which is at approximately the point of maximum collapse. The different traces correspond to the inviscid model, and the viscous model with different values of μ_{water}. In frame (A) the main difference between the inviscid and viscous models is that the jet on the underside of the bubble is more advanced in the inviscid model. On closer inspection, the shapes of the bubbles using the viscous model can be seen to be more uneven than in for the

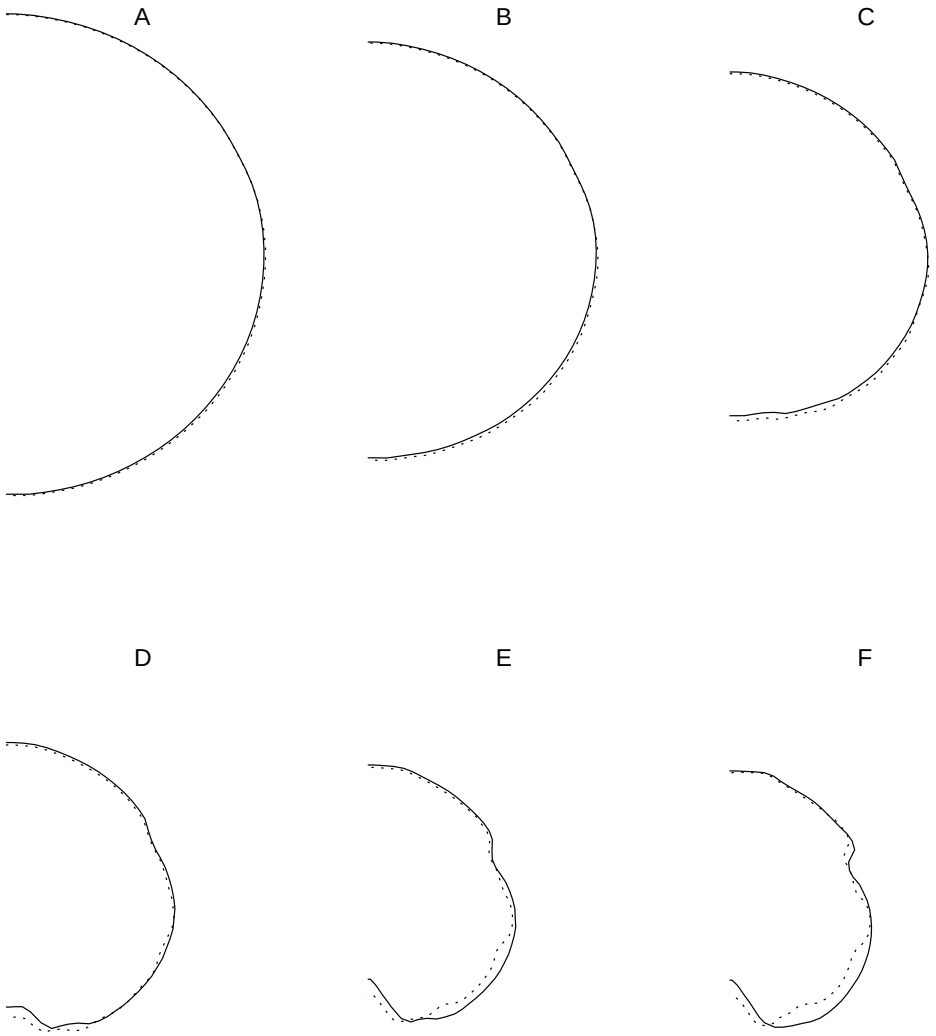


Figure 7.6: The shape of the bubble at different times during collapse, both with (dashed line) and without (solid line) the inclusion of viscosity. (A) $t = 62.7$ms; (B) $t = 65.3$ms; (C) $t = 67.7$ms; (D) $t = 69.9$ms; (E) $t = 71.8$ms; (F) $t = 73.7$ms.

inviscid model. For $\mu_{\text{water}} = 100\text{kgm}^{-1}\text{s}^{-1}$ (green trace) the bubble shape is different to all other traces, with more pronounced ripples on the underside of the bubble. It should

be noted that this value of μ_{water} is an increase from the true viscosity of ordinary water by a factor of 10^5.

In frame (B), whilst there are differences between all traces, there is a distinct qualitative difference between the trace for the invscid model (solid black line) and all others. The inviscid model gives a smoother surface during collapse, suggesting that viscosity destabilises interface *in this model*. The destabilising of the surface occurs even in the case where $\mu_{\text{water}} = 0$ and the viscous GFM is used (dashed red line). This effect is also present when the viscous GFM is used with $\mu_{\text{air}} = \mu_{\text{water}} = 0$ (not shown in Figure 7.7). Figure 7.7 does not show an increase in the magnitude of surface ''wrinkles'' as the value of μ_{water} is increased. This shows that the reduction in stability of the interface is not caused by the viscosity of the water, but by the application of the no-slip condition.

For instabilities due to the motion of an interface in the direction perpendicular to its plane, such as Rayleigh-Taylor instabilities, viscosity has a damping effect. However, when the velocity field around the interface has a non-zero component tangential to the interface, another type of instability can occur, due to the shear velocity between the two fluids. This is similar to the widely known Kelvin-Helmholtz instability (Tritton, 1988, p. 267). Without the no-slip condition, the model cannot capture shear flow instabilities at the air-water interface. This may be the reason for the decreased stability of the interface when the viscous GFM is used.

For the viscous model with $\mu_{\text{water}} \leq 10\text{kgm}^{-1}\text{s}^{-1}$, the jet which forms on the underside of the bubble is reduced compared with the viscous model. This effect is a complicated one, on which no clear conclusions can be drawn at this point. Intuitively, one would expect viscosity to damp phenomena such as jets, where there are strong velocity gradients transverse to the direction of motion. However, viscosity reduces the bubble rise rate, and hence the ambient pressure around the bubble is altered, which may also play a role. There are known errors associated with these jets due to the numerical instabilities inherent in the scheme, and these may also influence this result.

The effects observed in Figure 7.7 are interesting, but they perhaps say more about the code than the behaviour of physical bubbles. The interface is modelled in a more realistic way in the viscous model, and this is clear from Figures 7.2 to 7.5. The ability of the model to capture shear-flow instabilities in the viscous model is another improvement which makes the model more realistic. However, the two competing effects of viscosity on surface stability - damping Rayleigh-Taylor instabilities and causing shear-flow instablities - may balance differently in reality from the model. The computational mesh used in these simulations is coarse, and almost certainly influences the shapes the surface takes. Finally, it is important to remember that in reality the surface of the bubble is three-dimensional, whilst the simulation is two-dimensional. This may also qualitatively change the way in which viscosity influences the bubble shape.

At present, clusters of air guns - where the guns are located close enough together that the bubbles coalesce - are not usually investigated using models which calculate the details of the coalescence (Barker and Landrø, 2013), although this has been investigated using boundary integral methods (Cox *et al.*, 2004). The modelling of coalescing bubbles is a very difficult problem, due to the small length and time scales involved with thin surfaces which may be subject to high strain rates. Viscosity will certainly play a factor in coalescence. However, the results of the present simulations do not provide any insights into what effect viscosity may have on coalescing bubbles. A simulation using the NLAA artificial boundary condition and a finite volume scheme to model a cluster of air guns would have to be conducted in three dimensions. Furthermore, the model would need to account for surface tension for the results to be credible. Whilst such an investigation would be a valuable contribution, it is beyond the scope of the present work, and would require a finite volume scheme with more advanced interface modelling.

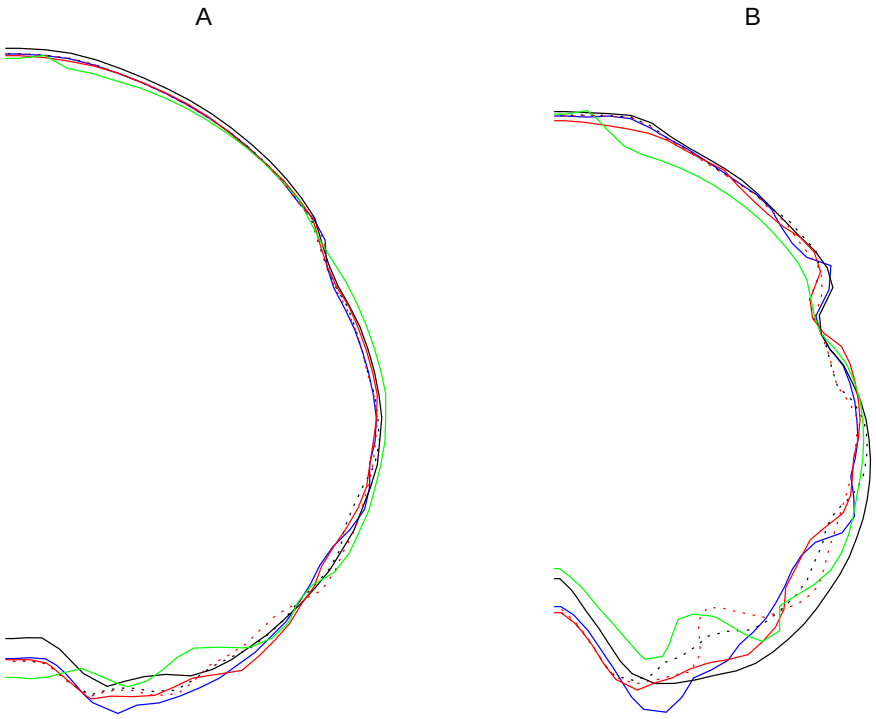


Figure 7.7: The shape of the bubble at (A) $t = 69.9$ms and (B) $t = 73.7$ms, for the inviscid model, and the viscous model for various different values of water viscosity. Solid black line - invscid model; dashed red line - $\mu_{\text{water}} = 0$; dashed black line - $\mu_{\text{water}} = 10^{-3}\text{kgm}^{-1}\text{s}^{-1}$; blue line - $\mu_{\text{water}} = 1\text{kgm}^{-1}\text{s}^{-1}$; red line - $\mu_{\text{water}} = 10\text{kgm}^{-1}\text{s}^{-1}$; green line - $\mu_{\text{water}} = 100\text{kgm}^{-1}\text{s}^{-1}$. In the inviscid model $\mu_{\text{air}} = 0$. In all other traces μ_{air} is set according to the equation of Sutherland (1893).

7.5.3 Bubble signature

Figure 7.8 shows the variation of interface position and pressure with time for the inviscid and viscous models. The traces in Figure 7.8 match very closely, to within 0.5% prior to $t = 0.07$ seconds. During the very last stages of collapse, the differences between the inviscid and viscous models increase. The viscous model shows marginally less

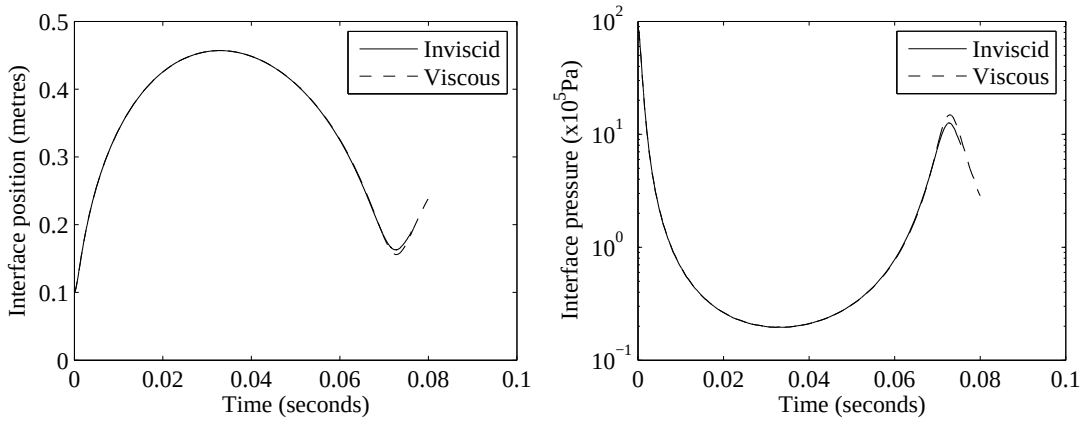


Figure 7.8: Variation of interface position (left) and interface pressure (right) with time with (dashed line) and without (solid line) viscosity.

GFM	μ_{air}	μ_{water}	P_{collapse} ($\times 10^6$Pa)
Inviscid	0	0	1.261
Inviscid	Sutherland	10^{-3}	1.283
Viscous	0	0	1.485
Viscous	Sutherland	10^{-3}	1.487
Viscous	Sutherland	0.489	1.529
Viscous	0	10^{-3}	1.486
Viscous	10^{-5}	10^{-3}	1.492
Viscous	Sutherland	1	1.514
Viscous	Sutherland	10	1.506
Viscous	Sutherland	100	1.354

Table 7.1: The peak pressure in the bubble during collapse, for varying values of viscosity, and different forms of the GFM. 'Sutherland' refers to the equation of Sutherland (1893) detailed in Section 7.3. Values of viscosity have units of kgm^{-1}s^{-1}.

damping than the inviscid model, with a higher peak pressure as the bubble collapses. This is in partial agreement with the results of Langhammer and Landrø (1993a), who observed a reduction in damping on collapse, and a reduction in the bubble period with increasing viscosity. The modelled results show no change in bubble period. Table 7.1

shows the effects of changing the viscosity on the peak pressure developed in the bubble during collapse. The change of the GFM from inviscid to viscous has a significant impact on the collapse pressure, whilst the values given to the viscosity in the air and the water appear to have little effect. The lack of influence of viscosity on bubble radius and bubble pressure is in agreement with the arguments of Taylor and Davies (1943) and Ziolkowski (1970). However, another potential reason for this is the numerical viscosity inherent in the scheme. If the effective viscosity of the scheme without the inclusion of viscous terms is greater than the actual viscosity, then the effects of the viscous terms will be minimal. Discussions of the dissipation mechanisms of Godunov-type schemes is given by Park and Kwon (2003) and Xu and Li (2001), who points out that the effective viscosity of Godunov-type schemes depends on the flow solution and the mesh construction, and is not necessarily consistent with the Navier-Stokes viscous terms. The final three rows of Table 7.1 show the effect on P_{collapse} as μ_{water} is increased far beyond its true value. There is little change in P_{collapse} for $\mu_{\text{water}} = 1$ and $10\text{kgm}^{-1}\text{s}^{-1}$. For $\mu_{\text{water}} = 100\text{kgm}^{-1}\text{s}^{-1}$, P_{collapse} is reduced significantly. This reduction in interface pressure on collapse only occurs within the last 2ms of collapse, as the jet begins to form. Prior to $t = 69$ms the P_{int} for $\mu_{\text{water}} = 10^{-3}\text{kgm}^{-1}\text{s}^{-1}$ and $\mu_{\text{water}} = 100\text{kgm}^{-1}\text{s}^{-1}$ agree to within 0.1%.

7.5.4 Rise rate

I calculate the vertical location of the centre of mass of the bubble z_b, as in Chapter 6, Section 6.5. Table 7.2 shows the distance risen by the bubble at $t = 68$ms, with both the viscous and inviscid versions of the GFM, with and without the viscous terms. The influence of the viscous terms on the distance risen is small, at 2.2%, whilst the influence of the viscous GFM on the distance risen is significant, at approximately 20%. Note that the variation in bubble shape due to surface instabilities observed in Figure 7.7 is contributes a small amount to the values of z_b. The drag on a body moving through a fluid is composed of two parts - form drag and skin friction drag. Form drag is due to the size and shape of the body, and the energy imparted on the fluid by the body as the fluid moves around the body. Skin friction drag is due to the friction of the fluid against

the surface of the body, which causes energy to be transferred from the body to the fluid, and is influenced by viscosity and the surface area of the body. For a bluff body such as a sphere, form drag dominates. However, the influence of skin friction drag is not negligible. The inviscid model is unable to capture skin friction drag on the bubble as it rises, as the no-slip condition is not applied at the interface, and the forces due to viscosity are not included. The viscous model captures the boundary layers on the bubble surface. The decrease in bubble rise rate in the viscous model is predominantly due to skin friction drag.

GFM	μ_{air}	μ_{water}	z_b (m)	% change
Inviscid	0	0	0.0270	0
Inviscid	Sutherland	10^{-3}	0.0264	-2.2%
Viscous	0	0	0.0215	-20.4%
Viscous	Sutherland	10^{-3}	0.0221	-18.1%
Viscous	Sutherland	1	0.0212	-19.7%
Viscous	Sutherland	10	0.0217	-17.8%
Viscous	Sutherland	100	0.0227	-14.0%

Table 7.2: The distance risen by the bubble at $t = 68$ms, for varying values of viscosity, and different forms of GFM. 'Sutherland' refers to the equation of Sutherland (1893) detailed in Section 7.3. Values of viscosity have units $\text{kgm}^{-1}\text{s}^{-1}$. The final column shows the percentage change in distance risen compared with the inviscid model in the first row.

7.6 Conclusions

I include viscous terms in my model, to solve numerically the full Navier-Stokes equations, and simulate a two-dimensional air gun bubble. To obtain viscous results, the code uses a version of the GFM which enforces the no-slip condition at the air-water interface, and includes viscous terms away from the interface using a second-order method. Results show viscosity changes the structure within and around the bubble, as the inclusion of viscosity demands boundary layers at the interface. Within the bubble,

viscosity has a significant effect on the motion, and the velocity field is noticably altered. However, far from the bubble viscosity has very little influence. During the latter stages of collapse the bubble surface is unstable, and the bubble shape is sensitive to viscous terms. The results show that the effect of viscosity on the signal emitted by the bubble is very small. The changes that do occur are in partial agreement with Langhammer and Landrø (1993a); I observe a reduction in damping on collapse. The no-slip condition enables shear-flow instabilities to be captured, causing the bubble shape to be more uneven during the late stages of collapse. The effect of viscosity on the bubble rise rate is significant, with modelled rise rates being lower when viscous terms are included, due to the no-slip condition. In order to quantify the effects of the viscous terms, the effective viscosity of the scheme must be quantified. A scheme with less inherent viscosity would be beneficial to further investigations.

Chapter 8

The effects of the sea surface

8.1 Introduction

In the previous chapters I consider an air gun bubble in water which extends infinitely in all directions. In reality, there are boundaries such as the sea surface, the sea floor, and various other obstacles. In typical air gun operation, the air gun is between 5 and 20 metres below the sea surface. The sea is usually deep enough that reflections from the sea floor either do not reach the bubble during the period of interest, or are weak enough to be neglected. However, the depth of the air gun is such that reflections from the sea surface cannot be neglected. For the purposes of air gun bubble modelling it is acceptable to assume the sea to be an infinite half-space, bounded above by the sea surface, neglecting any other boundaries, such as the sea floor.

Consider a bubble located at a depth d beneath the sea surface. The sea surface is smooth, and the density of the air above the surface is negligable. Provided the bubble is deep enough such that the velocity at the sea surface is small, the system can be modelled by removing the sea surface (allowing the sea to extend infinitely in all directions) and placing a 'ghost' bubble vertically above the actual bubble at a distance of $2d$. This configuration is shown in Figure 8.1. The pressure wave produced by the

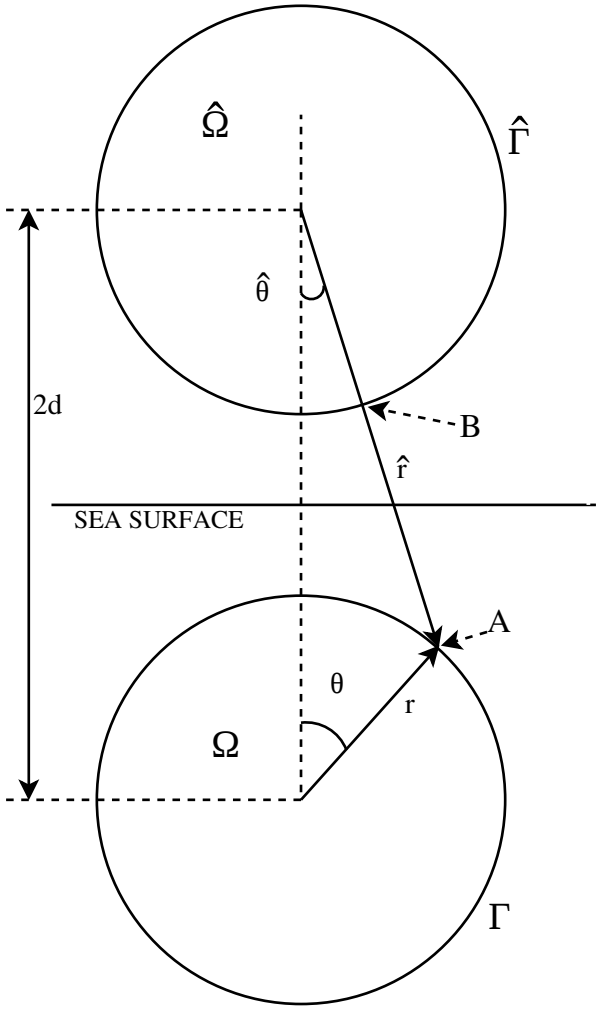


Figure 8.1: Schematic diagram of the configuration with the ghost. Note that the diagram is not to scale; in reality, $d \gg R_D$. Note also that the diagram shows the domains as full circles, whilst, due to the symmetry imposed, the computational regions are semicircles, containing the set of points $\mathbf{r}, \hat{\mathbf{r}} \in ([0, R_D], [0, \pi])$.

ghost bubble mirrors that produced by the real bubble, but with a reflection coefficient of -1. This approach of exploiting the symmetries of a problem in order to simplify its solution is commonplace in many branches of physics. This method has been used for many years to model single air guns and air gun arrays (Ziolkowski (1970), Ziolkowski *et al.* (1982), Parkes *et al.* (1984) and Ziolkowski (1998)).

In the past, air gun modelling has usually been based on homogeneous spherical bubbles, and the assumption that far from the bubble it can be considered a point source. Most

models describe bubbles in terms of a time varying pressure and radius only, and the effects of the sea surface on the bubble are included using a straightforward modulation of hydrostatic pressure terms (Ziolkowski (1970), Dragoset (1984), Laws *et al.* (1990), Landrø (1992), Langhammer (1994) and Ziolkowski (1998) and references therein). In the more complex two-dimensional models presented in Cox *et al.* (2004), the sea surface is only included in the results by superimposing the signals of the bubble and its image, and there is no simulation of the interaction of the ghost wave with the bubble. Data from near-field hydrophones show high frequencies when and shortly after the initial pulse due to the ghost bubble reaches the hydrophone.

The upper panel of Figure 8.2 shows a pressure measurement 1 metre from an air gun. This plot is taken with permission from Ziolkowski (1998), Figure A-6(a). Note what looks like high frequency noise at approximately 30 to 50ms as the signal due to the ghost interacts with the bubble. The lower panel of Figure 8.2 shows the pressure 1 metre from an air gun, simulated using a simple model described in Appendix D (using equations D.4 and D.5 with $n = 1.13$). The sea surface is included by simply superimposing the extrapolated wavefield due to the ghost over the wavefield propagating directly from the real bubble as in, for example, Cox *et al.* (2004). The axes in this plot have been scaled to provide a better comparison with the upper panel. The high frequencies can clearly be seen in the top panel of Figure 8.2, at approximately 30ms. The simple approach to account for the sea surface, the results of which are shown in the lower panel of Figure 8.2, does not show this region of high-frequency noise, as the effect of the sea surface is to add a scaled and delayed copy of the outgoing to wavefield. Results similar to this are shown in multiple figures in Cox *et al.* (2004). There is clearly a discrepancy between data and the approaches currently used in modelling.

In Chapters 3 and 4 I develop a two-dimensional finite volume model for air gun bubble simulations, including a new artificial boundary condition based on the non-linear acoustic approximation (NLAA) of Ziolkowski (1998). In Chapters 5 and 6 I demonstrate the capability of this model to capture asymmetrical features of the bubble, and and inhomogeneous motion within the bubble. To account for the effects

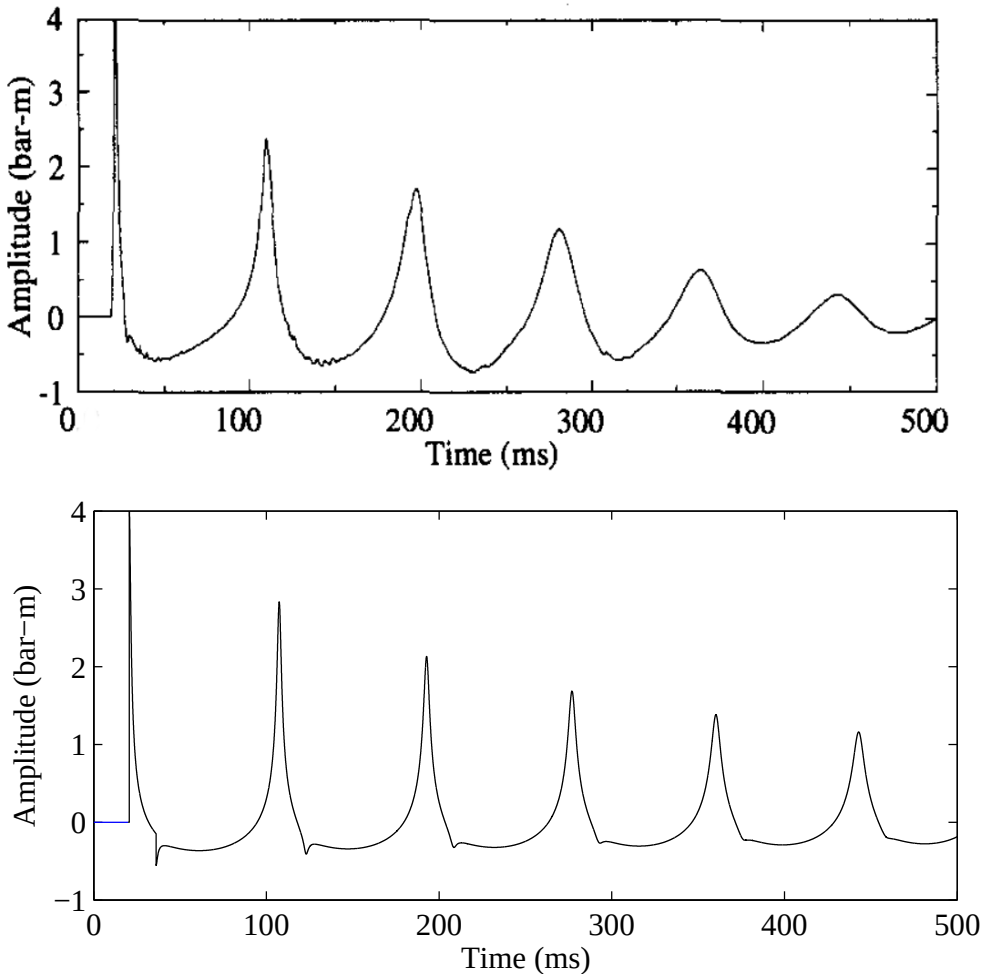


Figure 8.2: Upper panel: Pressure measurement 1 metre from an air gun, taken with permission from Ziolkowski (1998), Figure A-6(a). Lower panel: Pressure 1 metre from an air gun, obtained from simulations.

of the ghost bubble in the model, the wavefield produced by the ghost bubble must be somehow transmitted into the computational domain.

In this Chapter I present a method for including the effects of the sea surface in the new model. I use the NLAA to calculate the signal due to the ghost bubble at the boundary of the domain, and use this to augment the boundary condition developed in Chapter 3, thus transmitting the signal into the computational domain. I obtain numerical results using the computational implementation presented in Chapter 4, with the modified version of the NLAA boundary condition. The results show clearly that the method

succesfully transmits the ghost wave into the domain, and that it subsequently interacts with the bubble. This extension increases the practical value of the NLAA boundary condition. I believe these results to be the first air gun bubble simulations in which the interaction of the ghost wave with the bubble is modelled in detail, and in an asymmetric manner. I present data from experiments which supports this theory.

The layout of the chapter is as follows. In Section 8.2 I describe the method by which the signals due to the ghost are incorporated into the NLAA boundary condition. Section 8.3 contains the results and discussion of the present air gun model, including the effects of the ghost. In Section 8.4 I present data from experiments which supports my numerical results. Section 8.5 is a summary of conclusions.

8.2 Method for including the ghost

Figure 8.1 shows the configuration when the sea surface is included. Let Ω and Γ be the domain and the domain boundary, as in all previous chapters. Consider a ghost domain $\hat{\Omega}$, bounded by $\hat{\Gamma}$, which is a reflection of Ω in the sea surface, such that it is centred on the point $\left(r_{\hat{\Omega}_0}, \theta_{\hat{\Omega}_0}\right) = (2d,\, 0)$, where d is the depth of the gun below the sea surface. Note that whilst the depth of the bubble changes as the bubble rises, the computational domains are stationary. The position vector in a coordinate system based at the centre of $\hat{\Omega}$ is $\hat{\mathbf{r}} = \left(\hat{r}, \hat{\theta}\right)$, as shown in Figure 8.1. Note that the reflection of the domain leads the coordinate system based in $\hat{\Omega}$ to be left-handed, and that should the method be extended to three-dimensional simulations, care must be taken in this respect. Let the solution within Ω be $\mathbf{U}$, and the solution within $\hat{\Omega}$ be $\hat{\mathbf{U}}$. The solution within $\hat{\Omega}$ is assumed to behave identically to the solution within Ω, such that $\hat{\mathbf{U}}\left(\hat{r}, \hat{\theta}, t\right) = \mathbf{U}\left(r, \theta, t\right)$. The sea surface reflection coefficient of -1 is applied later.

The NLAA, developed by Ziolkowski (1998), can be used to calculate the velocity and pressure outside the bubble based on the past solution on the bubble wall. It is this approximation on which the boundary condition developed in Chapter 3 is based. I provide the derivation of this approximation in Chapter 3. I denote the approximate

solution calculated with the NLAA, based on the solution $\mathbf{U}$ on Γ, $\mathbf{U}_\Gamma$, as $\mathbf{U}_{NLAA}$. The approximate solution calculated with the NLAA, based on the solution $\hat{\mathbf{U}}$ on $\hat{\Gamma}$, $\hat{\mathbf{U}}_{\hat{\Gamma}}$, is $\hat{\mathbf{U}}_{NLAA}$. I modify the NLAA boundary condition, such that the solution external to Ω and $\hat{\Omega}$ is composed of the superposition of $\mathbf{U}_{NLAA}$ and $\hat{\mathbf{U}}_{NLAA}$. This external solution is used to define characteristics incoming to Ω, and so apply a boundary condition on Γ. In this way the waves due to the ghost are transmitted into Ω, where they subsequently interact with the bubble.

The method for including the ghost in the NLAA boundary condition is summarised as follows. The shortest distance between $\hat{\Omega}$ and each point on Γ is calculated. This is then used to provide a delay, τ, based on the constant and uniform speed of sound in the water, which is the time required for signals due to the ghost to reach Γ. The past solution on Γ is then searched to find the solution at $t - \tau$. The NLAA is then used to calculate the pressure and velocity, and their derivatives, on Γ due to signals emitted from $\hat{\Omega}$. This involves a non-linear scaling of the past solution. Once the pressure and velocity and their derivatives on Γ due to the ghost have been calculated, a geometrical transform is used to rotate these velocities and derivatives into a frame of reference oriented with the coordinate system in Ω. These are then used to augment the NLAA boundary condition described in Chapter 3.

I now give a detailed description of the method. The boundary condition is applied in this way for every cell on Γ. I describe the implementation in a cell A, which lies on Γ. Let the coordinates of cell A be $\mathbf{r_A} = (r, \theta)$. As A is on Γ, $r = R_D$, although for clarity of exposition this sustitution is not made in the following derivation. Denote the coordinates of cell A in the coordinate system of $\hat{\Omega}$ as $\hat{\mathbf{r}}_\mathbf{A} = \left(\hat{r}, \hat{\theta}\right)$. The unit vectors in the ghost-centered coordinate system are $\mathbf{e}_{\hat{\mathbf{r}}}$ and $\mathbf{e}_{\hat{\theta}}$. Simple pythagorean geometry leads to expressions for $\hat{r}$ and $\hat{\theta}$,

$$\hat{r} = \left(4d^2 - 4dr\cos\theta + r^2\right)^{\frac{1}{2}}, \tag{8.1a}$$

$$\hat{\theta} = \tan^{-1}\left(\frac{\sin\theta}{\left(\frac{2d}{r} - \cos\theta\right)}\right). \tag{8.1b}$$

Label as cell B the cell which contains the point at which a line between $(2d, 0)$ and A intersects $\hat{\Gamma}$, as in Figure 8.1. If $d \gg R_D$, $\hat{\theta}$ is small, and $\frac{\partial\hat{\theta}}{\partial\theta}$ is also small, such that in

some cases the index of cell B will not change as θ varies between 0 and π. Note that for $d/R_D = 7.7$, the maximum value of $\hat{\theta}$ is approximately 0.065 radians, or 3.7 degrees. In the present scheme, where $\delta\theta = \pi/50$, the index of cell B varies only between two adjacent cells as θ varies between 0 and π. The speed of sound is used to calculate the time taken for signals to travel from B to A, $\tau = (\hat{r} - r)/c$, where c is the speed of sound used in the NLAA calculations, based on the density and pressure at A. I define $t_G = t - \tau$, and denote the properties $(\cdot)$ at B at time t_G, $(\cdot)_{B,t_G}$. Then ρ_{B,t_G}, u_{B,t_G}, p_{B,t_G} and $\frac{\partial u_{B,t_G}}{\partial t}$ are used to calculate the wavefunction and its derivatives, f, f' and f'', using equations 3.13, 3.14 and 3.15 (Chapter 3, Section 3.2)

$$f = r^2 \left(u_{B,t_G} - \frac{p_{B,t_G} - p_\infty}{\rho_\infty c} - \frac{u_{B,t_G}^2}{2c} \right), \tag{8.2a}$$

$$f' = r \left(\frac{p_{B,t_G} - p_\infty}{\rho_\infty} + \frac{u_{B,t_G}^2}{2} \right), \tag{8.2b}$$

$$f'' = rc\frac{\partial u_{B,t_G}}{\partial t} - c\frac{p_{B,t_G} - p_\infty}{\rho_\infty} - \frac{cu_{B,t_G}^2}{2}. \tag{8.2c}$$

Note that this step is actually done separately. At each time step, f, f' and f'' are calculated for all B required. This is the only historical part of the solution which is stored. This approach minimises the storage requirements of the code. If $t_G < 0$, the signal due to the ghost cannot have reached Ω. In this case, the NLAA boundary condition is simply applied in the usual manner, as described in Chapter 3, whilst f, f' and f'' are stored for use once $t_G \geq 0$. Note that when searching the past solution for the solution at t_G, the nearest sample is used. A better method would be to use interpolation. However, as the time step is typically of the order of 10^{-6} the error introduced by this approach is small, and in any case the consequence will only be the introduction of a small amount of dissipation in the ghost wave which is transmitted into the domain.

The NLAA is used to estimate the solution at A due to the past solution at B. Let $\hat{u}_G$, p_G, $\frac{\partial \hat{u}_G}{\partial \hat{r}}$ and $\frac{\partial p_G}{\partial \hat{r}}$ be the velocity component in the direction $\mathbf{e}_{\hat{\mathbf{r}}}$ due to the ghost, the pressure due to the ghost, and their derivatives, at A. From Chapter 3, equations 3.9, 3.14, 3.10 and 3.12, they are given by

$$\hat{u}_G = R_{ss} \left\{ \frac{f}{\hat{r}^2} + \frac{f'}{\hat{r}c^2} \right\}, \tag{8.3}$$

$$p_G = R_{ss}\rho_\infty \left(\frac{f'}{\hat{r}} - \frac{\hat{u}_G^2}{2} \right) + p_\infty, \tag{8.4}$$

$$\frac{\partial \hat{u}_G}{\partial \hat{r}} = R_{ss} \left\{ \frac{-2f}{\hat{r}^3} - \frac{2f'}{\hat{r}^2 c} - \frac{f''}{\hat{r}c^2} \right\} \tag{8.5}$$

and

$$\frac{\partial p_G}{\partial \hat{r}} = -R_{ss}\rho_\infty \left\{ \hat{u}_G \frac{\partial \hat{u}}{\partial \hat{r}} + \frac{f'}{\hat{r}^2} + \frac{f''}{\hat{r}c} \right\}, \tag{8.6}$$

where R_{ss} is the reflection coefficient of the sea surface, and $R_{ss} = -1$. If the sea surface were replaced with a rigid boundary, this would be accounted for in the model by setting $R_{ss} = 1$. I neglect $\hat{v}_G$, $\frac{\partial \hat{u}_G}{\partial \hat{\theta}}$ and $\frac{\partial p_G}{\partial \hat{\theta}}$, as the NLAA is based on the assumption of spherical symmetry. Let $\psi = \pi - \hat{\theta} - \theta$. The components of the velocity field due to the ghost in the coordinate system centred on the real bubble can be calculated using the rotation $\mathbf{u}_G = \mathbf{M}\left(\hat{u}_G, 0\right)^T$, where $\mathbf{u}_G = u_G \mathbf{e_r} + v_G \mathbf{e}_\theta$ and

$$\mathbf{M} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}, \tag{8.7}$$

giving

$$u_G = \hat{u}_G \cos\psi \quad \text{and} \quad v_G = \hat{u}_G \sin\psi. \tag{8.8}$$

The derivatives of $\hat{u}_G$ and p_G in the coordinate system centred on the real bubble can be obtained, again using the rotation $\mathbf{M}$, as $\nabla(.) = \mathbf{M}\hat{\nabla}(.)$, where $\nabla(.) = \frac{\partial(.)}{\partial r}\mathbf{e_r} + \frac{1}{r}\frac{\partial(.)}{\partial \theta}\mathbf{e}_\theta$ and $\hat{\nabla}(.) = \frac{\partial(.)}{\partial \hat{r}}\mathbf{e_{\hat{r}}} + \frac{1}{\hat{r}}\frac{\partial(.)}{\partial \hat{\theta}}\mathbf{e}_{\hat{\theta}}$. Hence the derivatives of p_G are given by

$$\frac{\partial p_G}{\partial r} = \frac{\partial p_G}{\partial \hat{r}} \cos\psi, \tag{8.9a}$$

$$\frac{\partial p_G}{\partial \theta} = r\frac{\partial p_G}{\partial \hat{r}} \sin\psi, \tag{8.9b}$$

and the derivatives of $\hat{u}_G$ are

$$\frac{\partial \hat{u}_G}{\partial r} = \frac{\partial \hat{u}_G}{\partial \hat{r}} \cos\psi, \tag{8.10a}$$

$$\frac{\partial \hat{u}_G}{\partial \theta} = r\frac{\partial \hat{u}_G}{\partial \hat{r}} \sin\psi. \tag{8.10b}$$

Because $2d \gg R_D$, the variation of $\hat{\theta}$ with r and θ can be neglected, allowing the approximation that $\frac{\partial \cos\psi}{\partial r} = \frac{\partial \sin\psi}{\partial r} = 0$, $\frac{\partial \sin\psi}{\partial \theta} = -\cos\psi$ and $\frac{\partial \cos\psi}{\partial \theta} = \sin\psi$. The errors introduced by this approximation will vary around Γ. The maximum relative value of

these errors is of the order of $R_D/2d$. The derivatives of the velocity field due to the ghost are obtained using equations 8.8 and 8.10

$$\frac{\partial u_G}{\partial r} = \frac{\partial \hat{u}}{\partial \hat{r}} \cos^2 \psi, \tag{8.11a}$$

$$\frac{\partial u_G}{\partial \theta} = r \frac{\partial \hat{u}}{\partial \hat{r}} \sin \psi \cos \psi + \hat{u} \sin \psi, \tag{8.11b}$$

$$\frac{\partial v_G}{\partial r} = \frac{\partial \hat{u}}{\partial \hat{r}} \cos \psi \sin \psi, \tag{8.11c}$$

$$\frac{\partial v_G}{\partial \theta} = r \frac{\partial \hat{u}}{\partial \hat{r}} \sin^2 \psi - \hat{u} \cos \psi. \tag{8.11d}$$

The results of equations 8.9 and 8.11 can be used to calculate characteristic waves incoming to A due to the ghost bubble, according to

$$\mathscr{L}_{1,G} = (u - c) \left\{ \frac{\partial p_G}{\partial r} - \rho_\infty c \frac{\partial u_G}{\partial r} \right\}, \tag{8.12a}$$

$$\mathscr{L}_{2,G} = -u \frac{\partial p_G}{\partial r}, \tag{8.12b}$$

$$\mathscr{L}_{3,G} = u \frac{\partial v_G}{\partial r}. \tag{8.12c}$$

Note that the wave speeds for $\mathscr{L}_i$ are still determined by the internal (to Ω) solution on Γ. The terms calculated in equations 8.12a to 8.12c are added to the values for $\mathscr{L}$ obtained using the NLAA boundary condition described in Chapter 3, and the pressure due to the ghost is used to modulate p_∞, according to

$$p_\infty = p_\infty \left(t = 0\right) + \frac{R}{\rho_\infty} \left(\frac{f'}{\hat{r}} - \frac{\hat{u}_G^2}{2} \right). \tag{8.13}$$

This completes the process by which the ghost bubble is included. All remaining aspects of the model are unaltered.

The above method is described for a virtual bubble located at $(2d, 0)$. However, with minor changes to the geometry, the method could be extended for a virtual bubble, or indeed another real bubble, at any location, or multiple virtual bubbles. Note that the imposition of axisymmetry limits the present model. A ghost bubble located at $(2d, \pi/2)$ would actually approximate a toroidal bubble with major radius d. The solution in $\hat{\Omega}$ does not have to mirror the solution in Ω. The two domains could be separate finite volume simulations of bubbles, and the above method would allow them to interact, without the need to simulate the entire region between the two bubbles. With three-dimensional finite volume models, this approach could be used to simulate entire arrays

of air guns, including the effects of the sea surface. However, this approach would be computationally expensive.

8.3 Numerical Results

All results in this chapter are obtained using the inviscid model described in Chapters 3 and 4, with the alterations to the NLAA boundary condition described in the previous section. Viscous terms are not included, and the GFM used is the inviscid version described in Chapter 4, Section 4.6. The initial conditions are, as in Chapter 7, given by

$$(\rho,\, u,\, p,\, \gamma,\, p_c) = \begin{cases} (102,\, 0,\, 8.85 \times 10^6 + p_h,\, 1.4,\, 0) & \text{if } 0 \leq r \leq 0.1, \\ (1000,\, 0,\, p_w,\, 7.0,\, 3 \times 10^8) & \text{if } 0.1 < r \leq R_D, \end{cases} \tag{8.14}$$

where $p_w = p_{atm} + 1000gd + p_h$, $p_h = -\rho g r \cos\theta$, $p_{atm} = 1.01325 \times 10^5$ and d is the depth of the bubble. In all cases, $g = 9.81$ and the CFL number is 0.8. $\delta\theta = \pi/50$ and, except where explicitly stated otherwise, $\delta r = 0.02$ and $R_D = 1$.

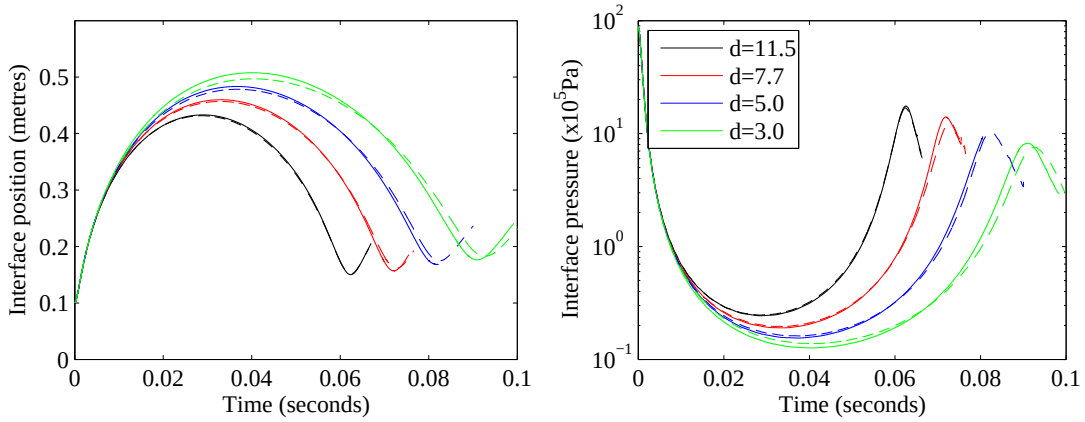


Figure 8.3: Variation of interface position and pressure with time, without the ghost (dashed lines), and with the ghost (solid lines), for bubbles at depths of $d = 11.5$, $d = 7.7$, $d = 5.0$ and $d = 3.0$.

I run the code with and without the ghost, for cases where the bubble is initially located at depths of $d = 11.5, 9.5, 7.7, 6.2, 5, 4, 3$ and 2.5 metres.

d (metres)	$\Delta \max(R_{int})_{rel}$	$\Delta P_{collapse,rel}$	$\Delta t_{collapse,rel}$
11.5	3.468×10^{-3}	3.737×10^{-2}	3.709×10^{-5}
9.5	4.822×10^{-3}	5.994×10^{-2}	-1.047×10^{-3}
7.7	6.616×10^{-3}	0.1288	-7.827×10^{-3}
6.2	8.471×10^{-3}	3.636×10^{-2}	-5.315×10^{-3}
5	1.042×10^{-2}	--	--
4	1.103×10^{-2}	7.736×10^{-2}	-1.185×10^{-2}
3	2.162×10^{-2}	8.68×10^{-2}	-1.896×10^{-2}
2.5	3.137×10^{-2}	0.2082	-2.535×10^{-2}

Table 8.1: The influence of the ghost on R_{int}, $P_{collapse}$ and $t_{collapse}$ as the depth is varied.

Figure 8.3 shows the time variation of R_{int} and P_{int} without the ghost (dashed lines), and with the ghost (solid lines) bubbles with depths $d = 11.5, 7.7, 5$ and 3 metres. The influence of the ghost on R_{int} and P_{int} is small for $d = 7.7$ and 11.5. When $d = 3$, the influence of the ghost is much more significant. In all cases, the effect of the ghost is to increase the maximum R_{int} and to reduce the minimum value of R_{int} reached during collapse. The maximum pressure in the bubble during collapse $P_{collapse}$ is also increased due to the ghost, and the bubble period, $t_{collapse}$ is decreased. As the wave due to the ghost passes the bubble, the pressure surrounding the bubble is reduced. This reduces the effective stiffness of the water to the expanding bubble, causing the larger peak R_{int} and the stronger, sharper collapse observed in Figure 8.3. Table 8.1 shows the relative changes $\Delta(\cdot)_{rel}$ in maximum interface position, maximum interface pressure during collapse and bubble period caused by the ghost, where $\Delta(\cdot)_{rel} = [(\cdot)_G - (\cdot)_{NG}] / (\cdot)_{NG}$, and a subscript G indicates the ghost is included, whilst a subscript NG indicates the ghost is not included. For $d = 5$ the final two columns are blank, because the simulation breaks down just prior to collapse when the ghost is included. This is simply an unfortunate consequence of the sensitivity of the bubble shape during collapse, and the sensitivity of the axial jets discussed in Chapter 6, Section 6.6, which cause the simulation to break down. For bubbles deeper than approximately 5 metres, there is a first-order convergence of $\Delta \max(R_{int})_{rel}$ with increasing d. For the shallower bubbles

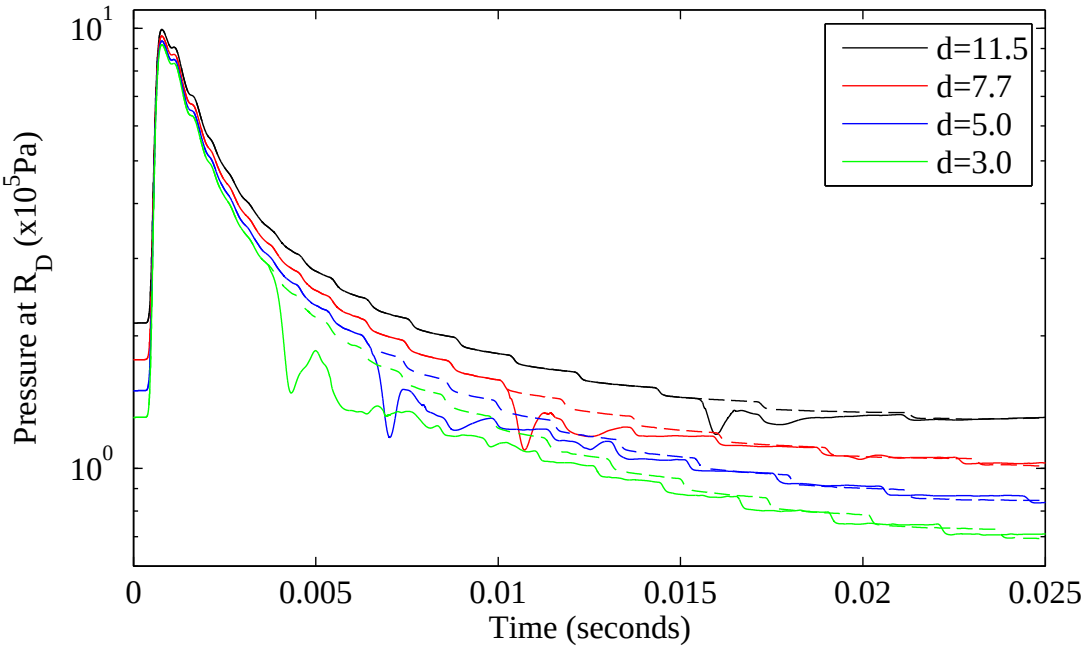


Figure 8.4: Variation of pressure on Γ at $\theta = \pi/2$ with time, without the ghost (dashed lines), and with the ghost (solid lines), for bubbles at depths of $d = 11.5, 7.7, 5.0$ and 3.0.

the effects of the ghost on the bubble radius appear to increase in proportion to $1/d^3$. The changes in the bubble properties on collapse due to the ghost do not show such strong trends. As d increases, $\Delta P_{collapse,rel}$ decreases. For all depths except $d = 11.5$ the ghost decreases the bubble period $t_{collapse}$, and this effect is greater for smaller d. The results are noisy towards the end of collapse, and the increase in $t_{collapse}$ when $d = 11.5$ is small enough that it may be due to the noisy results. The lack of quantifiable trends in the properties at the end of collapse is yet further evidence of the sensitivity of the bubble during collapse due to the instability of the surface. Small changes in bubble size and shape as the ghost wave impacts on the bubble during expansion lead to large changes in properties during collapse. It must also be noted that the change in behaviour observed at about $d = 5$ is likely to occur at different values of d as the initial volume and pressure of the bubble are changed.

Figure 8.4 shows the variation of pressure at the point $(r, \theta) = (R_D, \pi/2)$, henceforth referred to as $P_{R_D,\pi/2}$, with time, without the ghost (dashed lines), and with the ghost (solid lines) for $d = 11.5, 7.7, 5$ and 3 metres. Figure 8.5 shows the difference in $P_{R_D,\pi/2}$

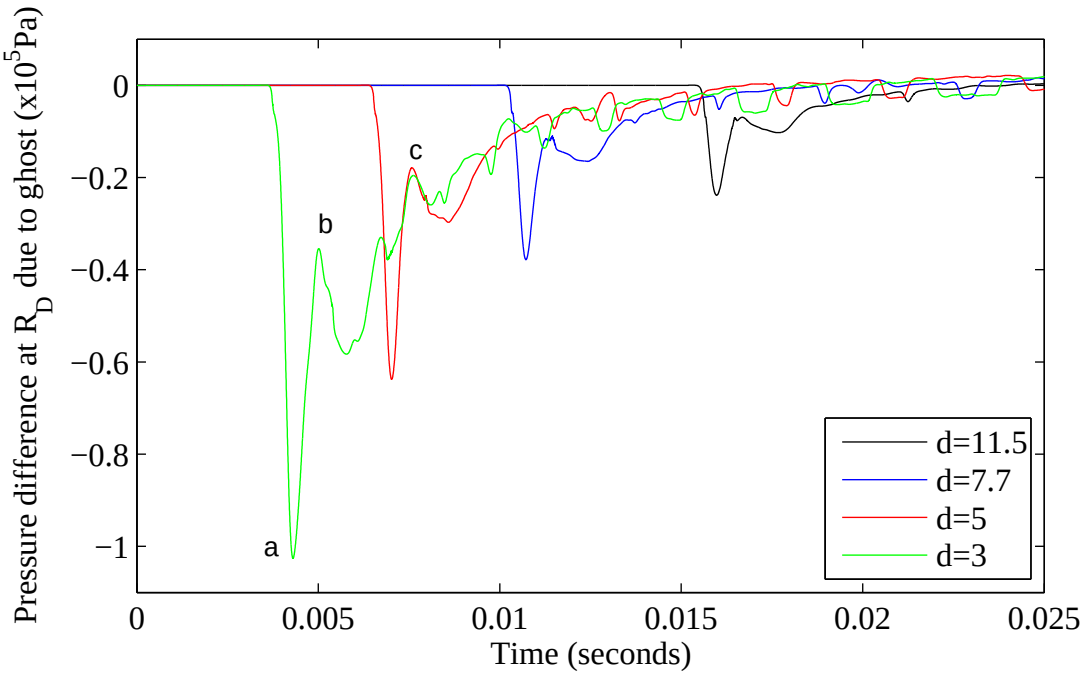


Figure 8.5: Variation of $\Delta P_{R_D,\pi/2}$ with time, for bubbles at depths of $d = 11.5, 7.7, 5.0$ and 3.0. $\Delta P_{R_D,\pi/2} = p\left(R_D, \pi/2\right)_G - p\left(R_D, \pi/2\right)_{NG}$ is the difference between the pressure at $(R_D, \pi/2)$ with and without the ghost.

with and without the ghost (ie. $P_{R_D,\pi/2,G} - P_{R_D,\pi/2,NG}$), for $d = 11.5, 7.7, 5$ and 3. The time at which the ghost begins to affect $P_{R_D,\pi/2}$ is proportional to d. The magnitude of the first disturbance to $P_{R_D,\pi/2}$ due to the ghost is proportional to $1/d$, which is to be expected, as according to the NLAA, the pressure around the bubble scales inversely with the distance from the bubble, in regions where the distance from the bubble is great enough that the velocity terms are small.

The steps in all traces in Figure 8.4 are due to the ghost fluid method (GFM), which is used to simulate the air-water interface. As the interface passes the centre of each cell, the set of cells which are interface cells changes, and so the conditions which control the transmission of energy and momentum across the interface change. This causes a small pressure pulse to propagate outwards from the interface. The notches after $t = 0.02$ seconds in all traces in Figure 8.5 are due to this aspect of the GFM. As the ghost causes a slightly different expansion rate, the times at which the interface passes a cell centre change, which leads to the notches visible in Figure 8.5. These notches

are not caused by the application of the boundary condition. This error is reduced by mesh refinement.

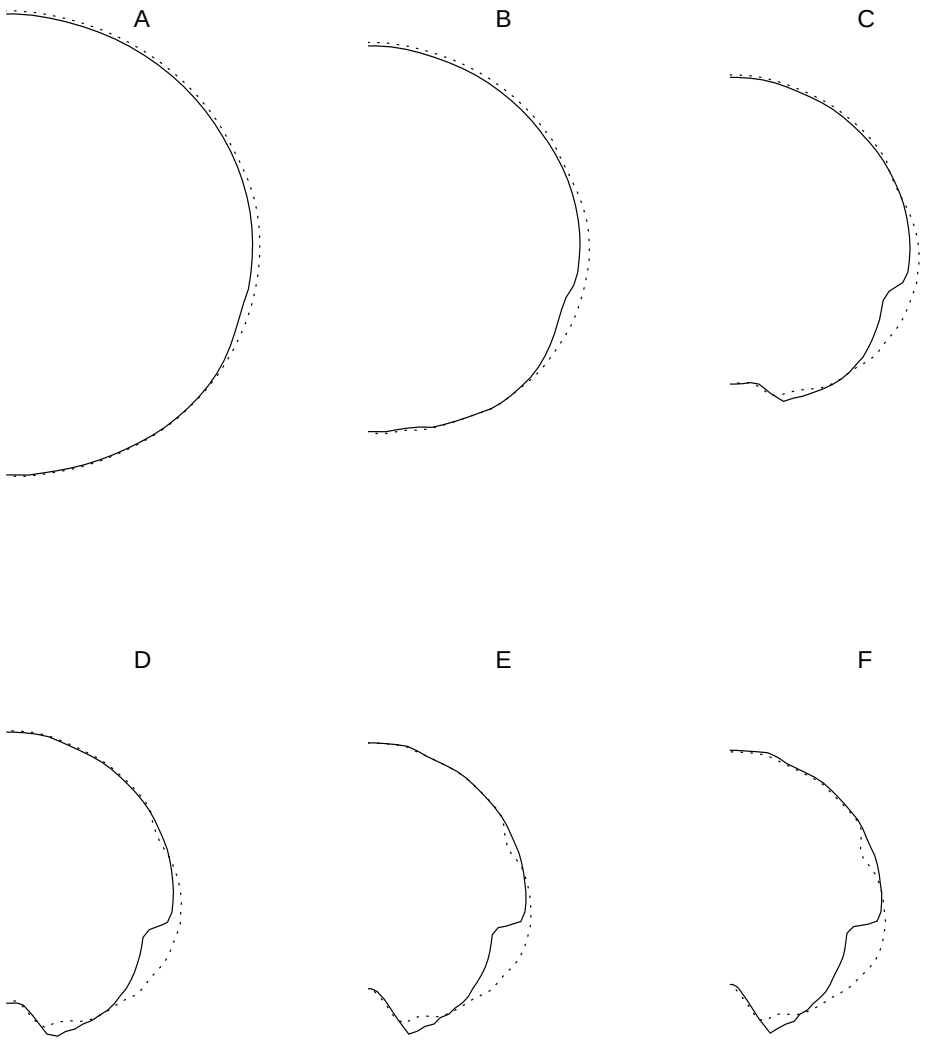


Figure 8.6: Bubble shapes at different times during collapse, without the ghost (dashed line) and with the ghost (solid line). $d = 7.7$ metres. (A) $t = 65.3$ms; (B) $t = 67.7$ms; (C) $t = 69.9$ms; (D) $t = 70.9$ms; (E) $t = 71.8$ms; (F) $t = 72.7$ms.

Figure 8.4 clearly shows a notch in the radiated pressure as the ghost wave passes.

Whilst I do not draw quantitative comparisons with experimental data or previous models, this is in qualitative agreement with, for example, Ziolkowski (1970), Ziolkowski and Johnston (1997), Ziolkowski (1998) and Cox *et al.* (2004). The ghost wave is calculated based on the past solution on Γ. The outgoing wave which impacts on Γ has been propagated across the domain, and due to the numerical viscosity inherent in the scheme, is slightly smeared. Hence the ghost wave also appears slightly smoother than in other models where there is no spatial discretisation. The incoming ghost wave has the same form as the initially outgoing pulse, but scaled and inverted. However, the results in Figures 8.4 and 8.5 do not show the ghost wave with this form. This is because the pressure which is recorded at $(R_D, \pi/2)$ is a combination of the wave which has travelled directly from the ghost to the point $(R_D, \pi/2)$, and the wave due to the ghost which has reflected off the bubble, a phenomenon which has not been captured in previous models. When the ghost wave impacts on the bubble it is partially transmitted into the bubble and partially reflected, with a reflection coefficient close to -1. The extra distance travelled by the wave which reflects off the bubble when it reaches $(R_D, \pi/2)$ is approximately R_D. The time between the trough, or first arrival (marked 'a' for $d = 3$ on Figure 8.5), and the peak, or second arrival (marked 'b' for $d = 3$ on Figure 8.5) of each trace in Figure 8.5 is close to R_D/c_w, where c_w is the speed of sound in water, which is evidence of this explanation. The signals produced by this mechanism ought to have a period of approximately $2R_{int}/c_b$, where c_b is the speed of sound in the bubble. For a bubble with $R_{int} = 0.27$ metres when the ghost wave arrives, and a speed of sound of approximately $c_b = 230\text{ms}^{-1}$, this gives an expected period of 2.3ms, and hence frequencies around 425Hz. The results shown in this chapter use a fairly dissipative scheme, and the polar coordinate system affects the passage of the ghost wave through the bubble. However, in Figure 8.5 the trace for $d = 3$ shows approximately this period (actually 2.48ms) between the second arrival (marked 'b') and the second reflection (marked 'c' on Figure 8.5). There is further evidence of this phenomenon visible in Figure 8.9, which I discuss later.

As mentioned above, towards the end of collapse, the shape of the bubble is sensitive to small disturbances. Figure 8.6 shows the shape of the bubble during collapse. The

bubble is initialised with $d = 7.7$ metres. The dashed line shows the shape of the bubble without the ghost, and the solid line shows the shape with the ghost. In frame (A), during the early stages of collapse, the bubble shapes are similar, although the bubble with the ghost is slightly larger than that without the ghost. There are slight differences in shape visible in frame (A), and with the ghost included the bubble is slightly less symmetrical. As the bubbles collapse, their shapes begin to differ more, as the slight variations in shape are amplified, and by the end of collapse (frame (F)) there are significant differences between the two bubble shapes. The main aspect of the ghost signal which influences bubble motion is the initial pulse of low pressure, which for $2d \gg R_D$ is close to planar, and passes through the bubble from above. It might be expected that this would cause the bubble to rise slightly, and for the upper surface to expand more rapidly than the lower surface. At the instant of impact, the speed of sound in the water is nearly an order of magnitude greater than the speed of sound in the air. Hence, the ghost wave travels round the bubble far faster than it propagates into the bubble, and so the asymmetric effect of the ghost wave on the bubble is reduced.

Figures 8.7 and 8.8 show the initial impact of the signal due to the ghost on the bubble, with $d = 7.7$ metres. Henceforth I refer to the first pulse of this signal as the ghost wave. To obtain these results, the model is run both with and without the ghost. The results without the ghost are subtracted from the results with the ghost to obtain the difference. In this way, the ghost wave can be seen. Let the subscript 'G' refer to results with the ghost present, and the subscript 'NG' to those without the ghost. Figure 8.7 shows the quantity $||\mathbf{u}_G| - |\mathbf{u}_{NG}||$ at various times as the ghost wave impacts on the bubble. Figure 8.8 shows the quantity $|(E_G - E_{NG})|/E_{NG}$, where E is the total energy, at the same instants in time. $|(E_G - E_{NG})|/E_{NG}$ is much greater in the air than in the water, so Figure 8.8 highlights the wave passing through the bubble.

Frame (A) shows the bubble just prior to the impact of the ghost wave. In frame (B) the ghost wave has just reached the bubble, and has begun to propagate both through and around it. Note that from frame (B) onwards, the ghost wave is not visible outside the bubble in Figure 8.8, as the relative changes in energy due to the ghost wave are

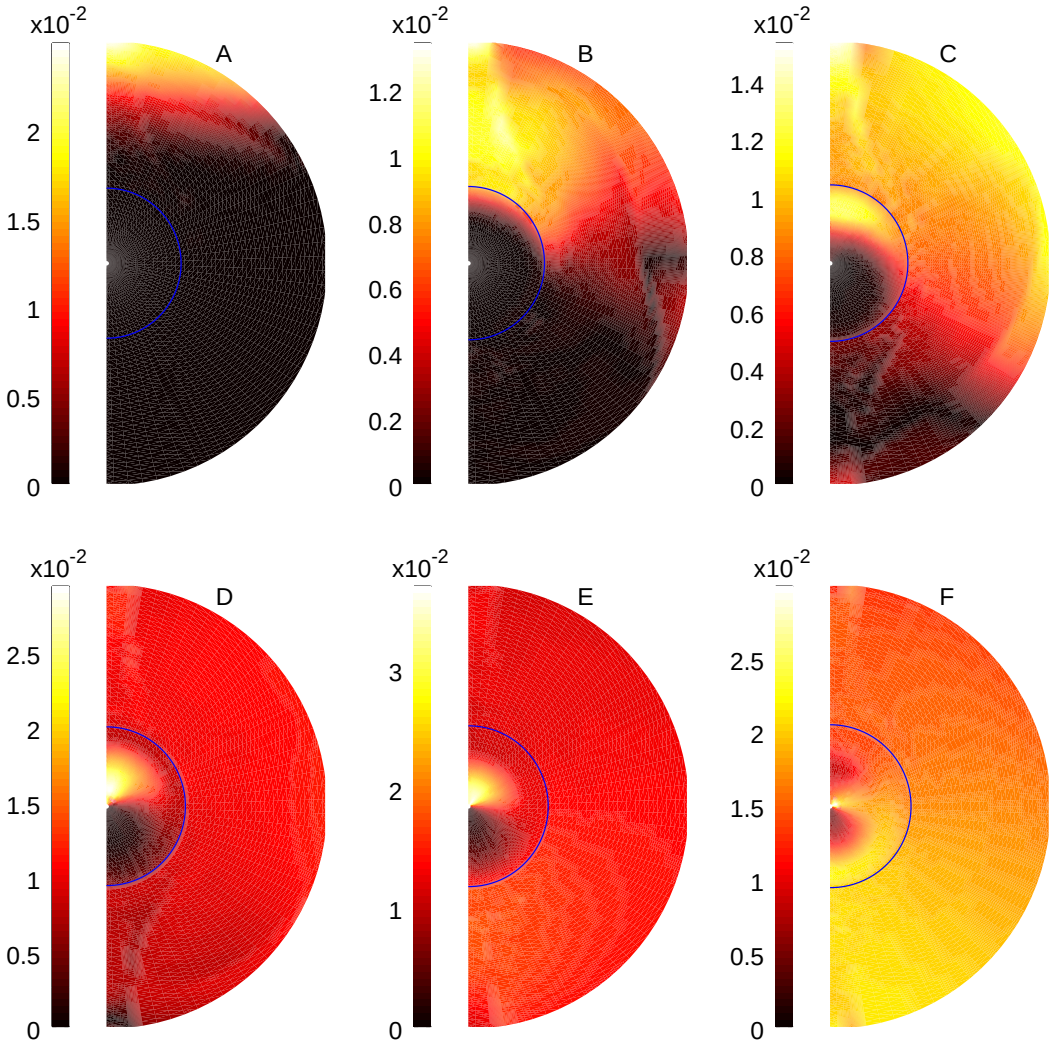


Figure 8.7: The ghost wave impacting on the bubble. The intensity of the image corresponds to the difference between the velocity magnitude with and without the ghost. The blue line represents the air-water interface. (A) $t = 10.21$ms; (B) $t = 10.81$ms; (C) $t = 11.41$ms; (D) $t = 11.86$ms; (E) $t = 12.31$ms; (F) $t = 12.76$ms.

much greater within the bubble than outside it. In frame (C) the ghost wave has propagated round the bubble, and is propagating into the bubble, as can be seen by the bright crescent in the upper half of the bubble in both figures. The difference in sound speeds within the bubble and in the water can now be clearly seen. Whilst the wave has travelled round the bubble and beyond in the water, a distance of more than two bubble radii, within the bubble it has not yet passed the origin, having travelled about

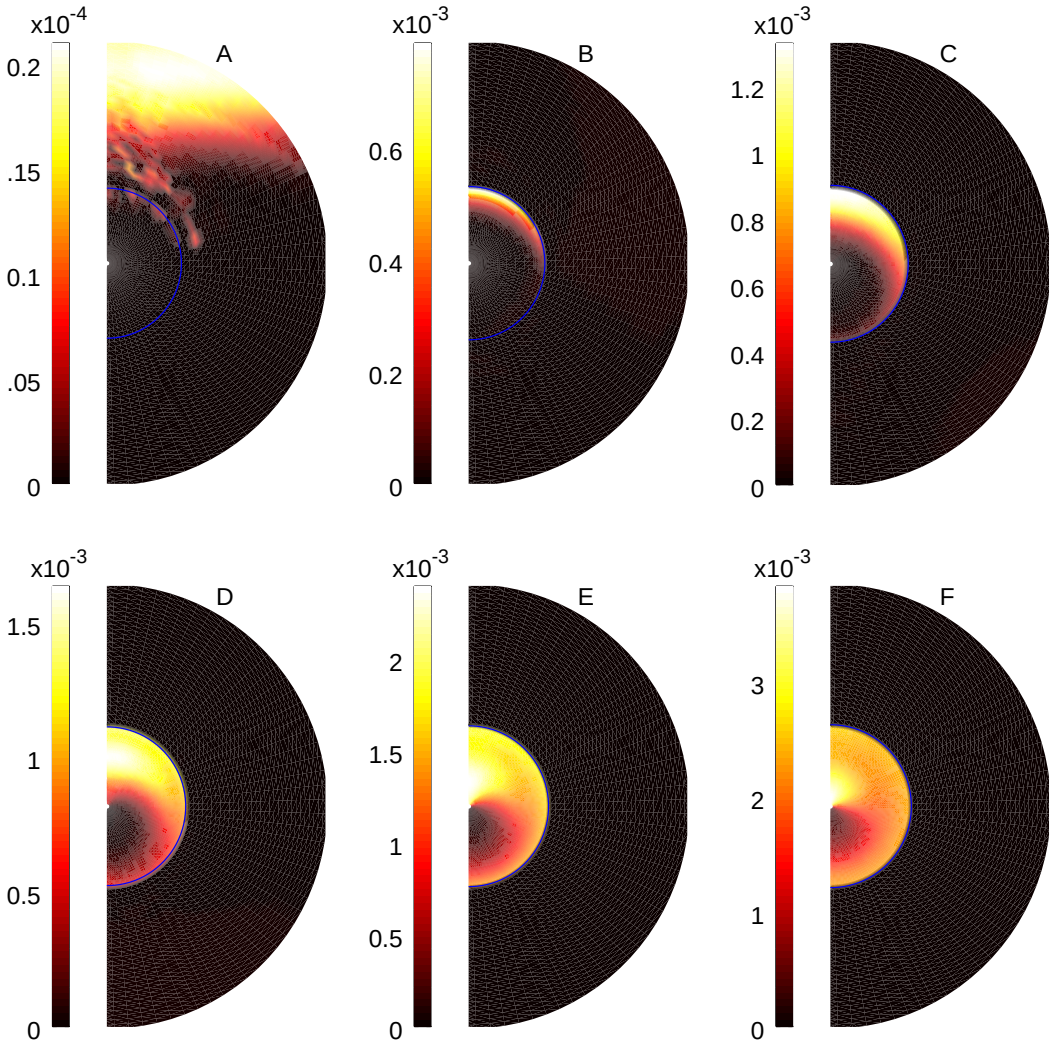


Figure 8.8: The ghost wave impacting on the bubble. The intensity of the image corresponds to the magnitude of the percentage difference in energy. The blue line represents the air-water interface. (A) $t = 10.21$ms; (B) $t = 10.81$ms; (C) $t = 11.41$ms; (D) $t = 11.86$ms; (E) $t = 12.31$ms; (F) $t = 12.76$ms. Note the different scale for frame (A).

0.4 bubble radii. In frames (D), (E) and (F) the ghost wave continues to propagate through the bubble, whilst in the water it has passed the bubble and continues out through the domain boundary. The ghost wave appears to get 'stuck' at the origin, and does not pass cleanly through it as expected. The reasons for this are two-fold. In part, the high sound speed in the water compared with the air means that the wave

propagating into the bubble is nearly spherical, and not planar. However, to some degree the behaviour close to the origin is a consequence of the polar coordinate system. A scheme in cylindrical coordinates would be beneficial to investigate this further.

As soon as the ghost wave impinges on the domain it affects the value of the time step, δt. In order to compare the velocity and energy fields, and to avoid complex interpolations, for these results I force the time step to be constant for $t > 10$ms. Despite this step, there are numerical artefacts present in Figures 8.7 and 8.8. These are visible as the two faint diagonal lines through the ghost wave in frames (A) to (C) of Figure 8.7, and the speckled areas in frame (A) of Figure 8.8. These are due to the subtraction of one field from another. The change in pressure due to the ghost causes small differences in the bulk motion in and around the bubble, and when one set of results is subtracted from the other, this leads to numerical artefacts mentioned above. This analysis can only be conducted at this stage, as the signal due to the ghost impacts on the bubble. Once the ghost has significantly changed the shape of the bubble and the flow field surrounding and within it, this analyis will give relatively meaningless results.

Finally I run the simulation at a higher resolution, with $\delta r = 0.01$, for $d = 7.7$ metres, both with and without the ghost. Figure 8.9 shows the impact of the ghost wave on the bubble in this case. Outside the bubble, the intensity of the image corresponds to the magnitude of the difference in pressure with and without the ghost, $|p_G - p_{NG}|$. Due to the difference in equations of state for air and water, the pressure differences within the bubble are small compared with those outside, and are not visible if the same scale is used. Within the bubble, contours of the difference in velocity magnitude are shown. Whilst there are no values on these contours, they clearly show the shape of the ghost wave propagating through the bubble.

Figure 8.9 shows several interesting features. The numerical artefacts of subtraction are clearly visible around the bubble - in the form of speckled regions - in all frames of Figure 8.9. These are not an error in the code, but simply an artefact of the method I employ to visualise the results. The diagonal line in frame (A) spreads round the domain as the ghost wave passes through. This line appears to originate from the

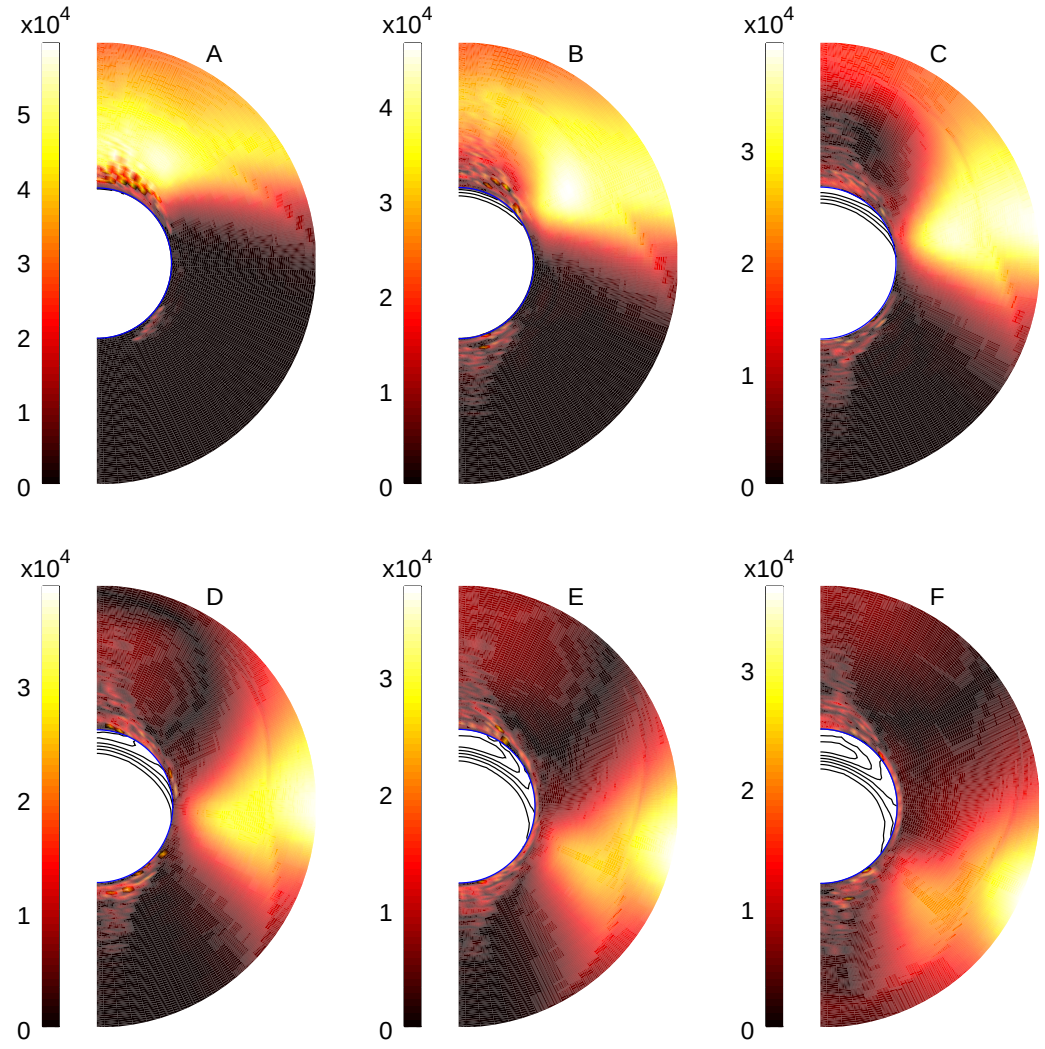


Figure 8.9: The ghost wave impacting on the bubble, modelled with $\delta r = 0.01$. The blue line represents the air-water interface. Outside the bubble the intensity of the image corresponds to the magnitude of the difference in pressure with and without the ghost. The contours within the bubble show the difference in velocity magnitude with and without the ghost. (A) $t = 10.48$ms; (B) $t = 10.63$ms; (C) $t = 10.78$ms; (D) $t = 10.93$ms; (E) $t = 11.08$ms; (F) $t = 11.23$ms.

boundary, and I believe it to be an error introduced by the boundary condition. Most importantly, Figure 8.9 shows the reflection of the ghost wave off the bubble. In frame (A) the ghost wave is just impacting on the bubble. In frame (B) the ghost wave has propagated a small distance into the bubble, and a reflection of the ghost wave (with

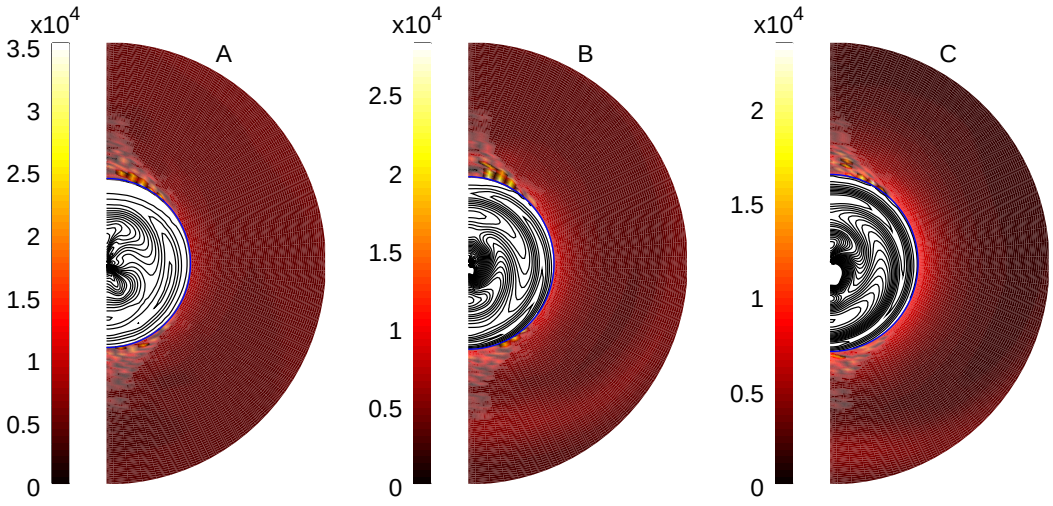


Figure 8.10: Plots showing the bubble after the ghost wave has impacted on the bubble, modelled with $\delta r = 0.01$. The blue line represents the air-water interface. Outside the bubble the intensity of the image corresponds to the magnitude of the difference in pressure with and without the ghost. The contours within the bubble show the difference in velocity magnitude with and without the ghost. (A) $t = 14.53$ms; (B) $t = 15.43$ms; (C) $t = 16.78$ms.

a reflection coefficient of -1) from the upper surface of the bubble is visible as the red patch above the bubble. This is even clearer a short time later in frame (C). Frames (D) to (F) show the ghost wave passing round the bubble, and being reflected from the air-water interface. It is this reflection which causes the higher frequencies observed in the pressure at $(R_D, \pi/2)$ in Figure 8.5. As a sound wave impacts on the bubble, it is partially transmitted through the interface, and partially reflected. The part of the ghost wave which propagates through the bubble will then be partially transmitted and partially reflected when it reaches far side of the bubble. The process repeats, with reducing amplitude, leading to higher frequencies in the pressure field around the bubble as the ghost wave passes the bubble. Figure 8.10 shows the solution a short time after the ghost wave has impacted on the bubble. The quantities plotted in Figure 8.10 are the same as those plotted in Figure 8.9. Concentric pressure waves which are nearly spherical can be seen propagating back and forth through the bubble. Every time these impact on the bubble surface, they are partially transmitted out into the water, as

can be faintly seen in frames (B) and (C). All these effects of the asymmetric bubble-ghost interaction have not been simulated in previous air gun modelling, and provide a potential explanation for the high frequencies observed as the ghost wave impacts the bubble in experimental measurements. In reality, the interaction is further complicated by the presence of the gun within the bubble.

8.4 Experimental Observations

In the previous section I describe a mechanism for bubble-ghost interactions not included in previous models. In this section I show measurements that support my theory. As a part of a Joint Industry Program (JIP), Petroleum Geo-Services (PGS) conducted the Svein Vaage broadband air gun study (Mattssonn *et al.*, 2012) between June and October 2007 and again between June 2009 and June 2010. In this study single air guns and air-gun clusters were fired at a test barge in a fjord on the west coast of Norway, whilst near-field and far-field signals were recorded. For the portion of data I analyse, a single 250 cubic inch ($4.1 \times 10^{-3}\text{m}^3$) air gun was suspected at a depth of 6 metres and fired at 2000psi (13.79MPa). A near-field hydrophone located a distance of 1.5 metres from the gun ports and at the same depth was used to record the signal. The data were recorded with a sample rate of 100kHz. I analyse a single shot of the gun, although the trends I observe are present in all shots with this configuration.

The left panel of Figure 8.11 shows the first 0.15 seconds of the signal recorded at the hydrophone. Note the ghost reflection at approximately $t = 0.014$ seconds, followed by small ripples until about $t = 0.04$ seconds. The right panel of Figure 8.11 shows the amplitude spectrum of this first 0.15 seconds of the signal. The small peak marked 'a' corresponds to frequencies between 400 and 600Hz, and is due to the ripples after the impact of the ghost wave.

The left panel of Figure 8.12 shows a portion of the data from Figure 8.11, around the time at which the ghost wave first reaches the hydrophone. The point marked 'a' corresponds to the first arrival of the ghost at the hydrophone. The point marked

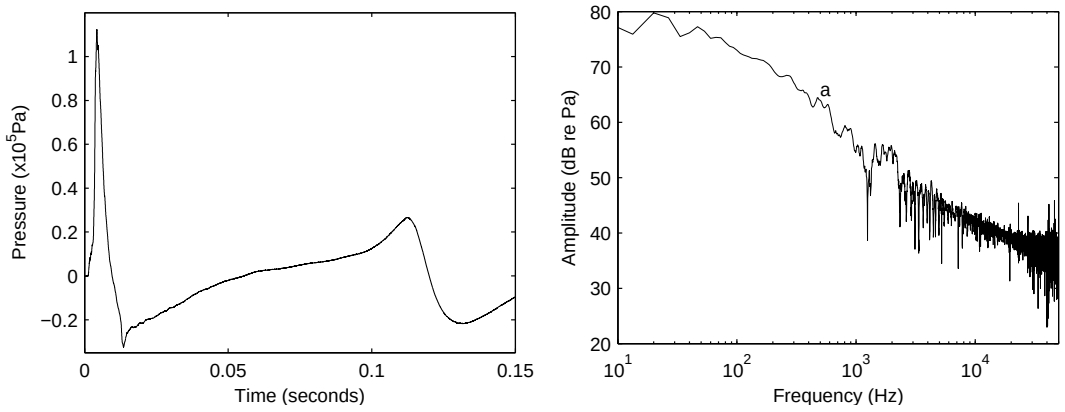


Figure 8.11: Near-field measurements from a 250 cubic inch air gun at 6 metres depth. Left panel - pressure measurement; right panel - amplitude spectrum.

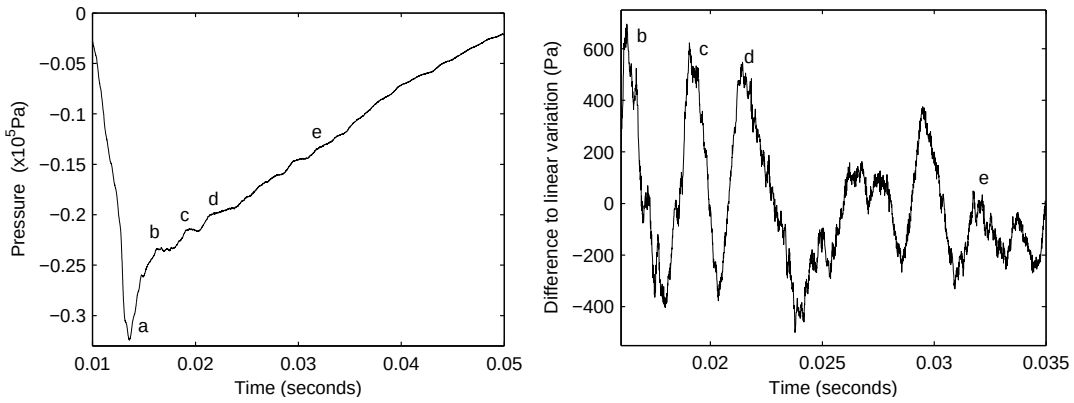


Figure 8.12: The portion of the near-field measurement around the time the ghost wave impacts the bubble. Left panel - pressure measurement; right panel - the difference between the direct measurement and a linear variation of pressure with time.

'b' corresponds to the arrival of the first reflection of the ghost wave off the bubble surface. The ripples, the first two of which are marked 'c' and 'd', correspond to the oscillation of the ghost wave within the bubble. These ripples continue in a slightly uneven manner, and with decreasing amplitude, until about $t = 0.04$ seconds. This is in good qualitative agreement with the theory developed in the previous sections, based on the numerical results shown in Figures 8.5, 8.9 and 8.10.

I calculate the average slope of the data in the left panel of Figure 8.12 between approximately the points marked 'b' and 'e'. By subtracting a straight line with this

slope from the measured values, and then subtracting the average of the result, I obtain an approximate image of these riples due to the ghost. I show this in the right panel of Figure 8.12. The points marked 'b' to 'e' correspond to those in the left panel. The frequencies of the large amplitude ripples vary between roughly 350 and 600Hz. The frequencies I observe here match those expected from the theory of ghost-bubble interactions put forward in the previous section. The ripples in the right panel of Figure 8.12 are not clean. There are other frequencies, both higher and lower, present in the data. In reality, the bubble contains a gun, and the presence of this influences the way the bubble focusses the ghost wave, and may introduce other frequencies as the effective size of the bubble differs in different directions. The high frequencies clearly visible in the right panel of Figure 8.12 are not present in the data prior to the impact of the ghost. Whilst not the focus of this paper, I note that these high frequencies may be due to the rattling of the gun, or the hydrophone, or mechanical resonances in the system as a whole.

8.5 Conclusions

The effects of the sea surface can be included by introducing an image of the bubble reflected in the sea surface, referred to as the ghost. I show how the ghost may be included in the model developed in Chapters 3 and 4. I extend the NLAA boundary condition to include the wavefield produced by the ghost in the approximation to the external solution, and transmit it into the domain, so that the interaction of the bubble with the ghost wave can be simulated.

The results show that the ghost causes slightly reduced damping in bubble oscillations. With the present initial conditions, for bubbles deeper than 5 metres, as the depth of the bubble is increased, the effect of the ghost on the maximum bubble radius varies with the inverse of the depth. For shallower bubbles, this effect varies with the inverse of the cube of the depth. The ghost strongly influences the final shape of the bubble, due to the instability of the surface during collapse. The present model is capable of capturing

the initial front due to the ghost wavefield impacting on the domain, and propagating through and round the bubble. The results show that when the ghost wave impacts on the bubble it is partially transmitted through the bubble, and partially reflected off the bubble. This reflection - and in reality several subsequent reflections - can expain the high frequencies present in measured bubble signatures as the ghost wave impacts on the bubble. Previous air gun models are unable to capture this phenomenon. This explanation of bubble-ghost interactions is supported by experimental data. In theory, measurements of the frequencies which occur after the impact of the ghost could be used to constrain models, and give some estimate of bubble size and composition.

In theory it is possible to use this approach to simulate an entire array of guns. An extension of this kind would require three-dimensional modelling, and a separate finite volume model for each bubble. The interactions between the different bubbles would follow the same method as the interaction between each bubble and its ghost. The computational cost of this approach would be significicant. The extension to the NLAA boundary conditions presented in this chapter significantly increases the practical value of the boundary condition, and further demonstrates its robustness in capturing highly asymmetric phenomena.

Chapter 9

Conclusions

In Section 9.1 I discuss and summarise the findings of this thesis. In Section 9.2 I make suggestions for, and speculate about, future work.

9.1 Findings

In this work I develop a new artificial boundary condition for use in finite volume simulations of oscillating bubbles. The boundary condition is designed for, but not limited to, the bubbles produced by seismic air guns. The boundary condition is based on the non-linear acoustic approximation (NLAA) of Ziolkowski (1998). The boundary condition is applied through the following process: (1) the solution of the finite volume scheme on the domain boundary is used with the NLAA to calculate an approximate solution external to the domain; (2) the spatial derivatives of the external solution are used to determine any characteristic waves which are incoming to the domain and (3) these characteristics may be used to update the solution on the boundary. The boundary condition is applicable in situations where the following assumptions are valid:

1. motion is *predominantly* in the radial direction;

2. motion is *roughly* spherically symmetric;

3. motion is of low Mach number ($Ma < 0.1$) on the boundary;

4. the viscous terms are *close to* zero.

If the words emphasised in the above set of assumptions are removed, then the assumptions on which the derivation of the boundary condition is based are obtained. In this thesis I demonstrate that the boundary condition performs well even when there are asymmetries in the motion and viscous forces are included in the finite volume scheme. If the errors incurred by the boundary condition are too great, the boundary can simply be moved further from the sources of asymmetry, vorticity or high Mach number flow, until the errors are reduced to an acceptable level.

The novel artificial boundary condition allows fixed grid finite volume simulations of oscillating bubbles on highly truncated domains. To the best of my knowledge, there are no published finite volume simulations of air gun bubbles. I develop a finite volume scheme in which to apply the boundary condition. It is based in a polar coordinate system and can be run in one or two dimensions. The scheme is capable of maintaining a sharp interface between two fluids, through the use of a ghost fluid method. The scheme uses piecewise constant reconstruction in each cell and HLLC fluxes. It is first-order accurate in time and space.

In one dimension, the errors due to the boundary condition show third-order convergence as the boundary is moved away from the source. The boundary condition performs well, yielding accurate results for underwater explosion problems, even in cases when the domain boundary is only 0.1% larger than the maximum bubble radius. The method allows long run time (10^5 time steps with a CFL number of 0.8) simulations on such highly truncated domains. The boundary condition gives good conservation properties (errors of less than 1%) over these very long run times. Compared with the approach of simply using a very large domain to avoid the difficulty of imposing an artificial boundary condition, the method can reduce computational costs by two orders of magnitude.

The two-dimensional air gun bubble simulations I present are capable of capturing complex asymmetric bubble phenomena, such as the deformation and translation of bubbles due to gravity, surface instabilities and the formation of jets. I show that these asymmetrical phenomena have a damping effect on the bubble motion. Two-dimensional results on a coarse mesh show second-order convergence of errors with increasing domain size, whilst with a refined mesh - comparable with that used in the one-dimensional simulations - the two-dimensional results show third-order convergence. The drop in convergence rate for coarse grids is due to a fundamental problem with the standard method of accounting for spherical coordinate systems. This is a problem which affects the simulations of other authors, although it is inadequately discussed in the literature.

The surface of an oscillating bubble is unstable during collapse. This phenomenon has been experimentally observed in a range of bubble types, including air gun bubbles. The two-dimensional model captures these instabilities as far as possible given the coarse computational mesh, although the numerical viscosity inherent in the scheme provides some damping to these instabilities. The two-dimensional results show that the final shape of the bubble obtained during collapse is highly sensitive to initial conditions, and to small disturbances throughout the oscillation. Whilst the instability of the surface is a Rayleigh-Taylor instability, the results show that the instability occurs at all times during collapse, contrary to the standard Taylor stability criteria.

When viscous terms are included in the model the results show different structures within and around the bubble, due to the no-slip constraint at the air-water interface. The model is capable of capturing the boundary layers around an air gun bubble. This is another phenomenon which previous air gun bubble models have not included. The introduction of viscosity has a significant effect on the motion within the bubble, whilst far from the bubble the motion is altered only a small amount. The effect of viscosity on the signal emitted by the bubble is very small. The changes that do occur are in partial agreement with Langhammer and Landrø (1993a), as a reduction in damping on collapse is observed. The no-slip condition allows the model to capture shear flow

instabilities at the bubble surface. The effect of viscosity on the bubble rise rate is significant, with modelled rise rates being lower when viscous terms are included.

I extend my boundary condition to include the effects of the sea surface. The solution external to the domain is approximated by the wavefield propagated from the domain, and from a mirror of the domain located around the ghost bubble. The approximate solution external to the domain is used to apply boundary conditions, and in this way the reflections from the sea surface are passed back into the domain and allowed to interact with the bubble. Whilst previous air gun models have simulated the interaction of the bubble with the sea surface via the adjustment of hydrostatic pressure terms, this is the first air gun bubble simulation capable of capturing the asymmetric effects of the bubble-ghost interaction.

The results show that the ghost causes slightly reduced damping in bubble oscillations. The effects of the ghost on the bubble scale with the inverse of the depth for deep bubbles (more than 5 metres), and with the inverse of the cube of the depth for shallow bubbles. The final shape of the bubble is strongly influenced by the ghost, due to the instability of the bubble surface during collapse. The results show that when the ghost wave impacts on the bubble it is partially transmitted through the bubble, and partially reflected off the bubble. This reflection - and in reality several subsequent reflections - explains the high frequencies present in measured bubble signatures as the ghost wave impacts on the bubble. This theory is supported by data from experiments. This explanation for the higher frequencies present in air gun signatures due to the sea surface is novel, and previous air gun models have not had the capability to capture this effect.

9.2 Future work

The results have demonstated that the boundary condition performs well, and most of the deficiencies in the present model are unrelated to the boundary condition, but are a consequence of the interface modelling, the coarse computational mesh, or the manner

in which the coordinate system is accounted for, or the coordinate system itself. Whilst the boundary condition is a valuable contribution, the numerical scheme in which it is implemented is a comparatively basic one. Whilst the model adequately demonstrates the efficacy of the boundary conditions, and is able to highlight some complex bubble phenomena, detailed investigations of aspects such as surface instabilities will require a more efficient and robust model.

There are several ways in which this could be achieved. A model which uses an irregular mesh, and perhaps an adaptive mesh, in order to resolve the complex shape of the interface in more detail would be a major improvement. This would reduce computational costs, and allow regions of interest to be better resolved. Polar coordinate systems contain a singularity along the polar axes and at the origin, and this is a major weakness of the current scheme. In cylindrical coordinate systems there is no singularity at the origin. There remains a singularity on the polar axis, although it is weaker. Using a cylindrical coordinate system would allow bubble rise and the ghost wave impact to be better captured. However, in cylindrical coordinates the boundary condition is unstable. A spherical domain with an irregular mesh, in which the mesh is polar on the outer boundary, cylindrical near the polar axis, and smoothly varies between these regions might give improvements. In order to use the method to include the effects of the sea surface to model an entire array, the simulation must be in three-dimensions. Whilst this would significantly increase computational costs, it would also allow the simulation of turbulence around the bubble and bubble surface instabilities.

The results of the present model provide further evidence for what has long been known, though infrequently admitted: air gun bubbles contain very complex phenomena, and an attempt to model all the details of an air gun bubble deterministically will inevitably be lacking in some aspects. Most previous air gun bubble models have assumed spherical homogeneous bubbles. Such models are inherently incapable of capturing the complex phenomena which have long been known to occur in air gun bubbles, some of which the present model has captured. However, aspects of the results obtained with the present model, such as the degree of damping caused by surface instabilities, or the effects of asymmetric bubble-ghost interactions, may in due course be empirically incorporated

into more simple air gun bubble models. Hence, whilst the present model is probably too computationally expensive to be used in the design of arrays, it may elucidate aspects of bubble behaviour which can be used to improve the day-to-day models used by industry.

Appendix A

Additional numerical procedures

A.1 WENO scheme

The following is a description of the fifth-order WENO-Z scheme due to Borges *et al.* (2008). I use this scheme to calculate spatial derivatives for the upwind scheme to solve the level set advection equation 4.11. Given a function f, approximated by values at locations $i-2$, $i-1$, i, $i+1$ and $i+2$, denoted f_j for $j = i-2...i+2$, this procedure calculates approximations to $f_{i-\frac{1}{2}}$ and $f_{i+\frac{1}{2}}$, $\hat{f}_{i-\frac{1}{2}}$ and $\hat{f}_{i+\frac{1}{2}}$. Three three-point stencils s_k, for $k = 0, 1, 2$, are used for interpolation. s_0 contains the points $(i-2, i-1, i)$, s_1 contains the points $(i-1, i, i+1)$ and s_2 contains the points $(i, i+1, i+2)$.

For each stencil k, approximations to $f_{i+\frac{1}{2}}$ and $f_{i-\frac{1}{2}}$, $f^k_{i+\frac{1}{2}}$ and $f^k_{i-\frac{1}{2}}$, are calculated using third-degree polynomials according to

$$f^k_{i+\frac{1}{2}} = \sum_{j=0}^{2} c_{k,j} f_{i-k+j} \tag{A.1a}$$

$$f^k_{i-\frac{1}{2}} = \sum_{j=0}^{2} c_{2-k,2-j} f_{i-k+j}, \tag{A.1b}$$

where the interpolation coefficients c are

$$\begin{aligned} c_{0,0} &= \tfrac{1}{3} \quad c_{0,1} = \tfrac{-7}{6} \quad c_{0,2} = \tfrac{11}{6} \\ c_{1,0} &= \tfrac{-1}{6} \quad c_{1,1} = \tfrac{5}{6} \quad c_{1,2} = \tfrac{1}{3} \\ c_{2,0} &= \tfrac{1}{3} \quad c_{2,1} = \tfrac{5}{6} \quad c_{2,2} = \tfrac{-1}{6} \end{aligned} \tag{A.2}$$

Expressed in full, these are

$$f^0_{i+\frac{1}{2}} = \frac{1}{3}f_{i-2} - \frac{7}{6}f_{i-1} + \frac{11}{6}f_i, \tag{A.3a}$$

$$f^1_{i+\frac{1}{2}} = \frac{-1}{6}f_{i-1} + \frac{5}{6}f_i + \frac{1}{3}f_{i+1}, \tag{A.3b}$$

$$f^2_{i+\frac{1}{2}} = \frac{1}{3}f_i + \frac{5}{6}f_{i+1} - \frac{2}{6}f_{i+2}, \tag{A.3c}$$

and

$$f^0_{i-\frac{1}{2}} = \frac{-1}{6}f_{i-2} + \frac{5}{6}f_{i-1} + \frac{1}{3}f_i, \tag{A.4a}$$

$$f^1_{i-\frac{1}{2}} = \frac{1}{3}f_{i-1} + \frac{5}{6}f_i - \frac{1}{6}f_{i+1}, \tag{A.4b}$$

$$f^2_{i-\frac{1}{2}} = \frac{11}{6}f_i - \frac{7}{6}f_{i+1} + \frac{1}{3}f_{i+2}. \tag{A.4c}$$

The final approximations, $\hat{f}_{i-\frac{1}{2}}$ and $\hat{f}_{i+\frac{1}{2}}$ are obtained through convex combinations of $f^k_{i+\frac{1}{2}}$, according to

$$\hat{f}_{i\pm\frac{1}{2}} = \sum_{k=0}^{2} w_k f^k_{i\pm\frac{1}{2}}, \tag{A.5}$$

where w_k are the weightings. In order to calculate the weightings, smoothness indicators β_k are calculated as

$$\beta_0 = \frac{13}{12}\left(f_{i-2} - 2f_{i-1} + f_i\right)^2 + \frac{1}{4}\left(f_{i-2} - 4f_{i-1} + 3f_i\right)^2, \tag{A.6a}$$

$$\beta_1 = \frac{13}{12}\left(f_{i-1} - 2f_i + f_{i+1}\right)^2 + \frac{1}{4}\left(f_{i-1} - f_{i+1}\right)^2, \tag{A.6b}$$

$$\beta_2 = \frac{13}{12}\left(f_i - 2f_{i+1} + f_{i+2}\right)^2 + \frac{1}{4}\left(3f_i - 4f_{i+1} + f_{i+2}\right)^2. \tag{A.6c}$$

The weightings are defined as

$$w_k = \frac{\alpha_k}{\sum_{j=0}^{2}\alpha_j}, \tag{A.7}$$

where

$$\alpha_k = \frac{d_k}{\beta_k^z} \tag{A.8}$$

and

$$\beta_k^z = \frac{\beta_k + \varepsilon}{\beta_k + \tau_5 + \varepsilon}, \tag{A.9}$$

in which $\tau_5 = |\beta_0 - \beta_2|$, and $\varepsilon = 10^{-40}$ is introduced to avoid division by zero. d_k are called the ideal weightings, and are

$$d_0 = 0.3, \qquad d_1 = 0.6 \quad \text{and} \quad d_2 = 0.1, \tag{A.10}$$

If $\beta_0 = \beta_1 = \beta_2$ the central fifth-order scheme for the full stencil is obtained. This reconstruction procedure can be implemented in an upwind scheme. Consider the function ϕ, which is governed by equation 2.1 (as in Chapter 2). I wish to obtain a high-order upwind approximation to $\partial\phi/\partial x|_i^n$. This can be achieved by setting $f_i^+ = \phi_i - \phi_{i-1}$ and $f_i^- = \phi_{i+1} - \phi_i$. The above WENO scheme is used to calculate $\hat{f}^+_{i-\frac{1}{2}}$ and $\hat{f}^-_{i+\frac{1}{2}}$. A fifth-order upwind approximation to $\partial\phi/\partial x|_i^n$ is then obtained by setting

$$\left.\frac{\partial\phi}{\partial x}\right|_i^n = \begin{cases} \frac{\hat{f}^+_{i-\frac{1}{2}}}{\delta x} & a > 0 \\ \frac{\hat{f}^-_{i+\frac{1}{2}}}{\delta x} & a \leq 0. \end{cases} \tag{A.11}$$

A.2 An approximate Riemann solver

The following is a description of the approximate Riemann solver, based on that due to Hu *et al.* (2009), used in all simulations carried out in this work.

The Riemann problem is defined by $R(\mathbf{U}_L, \mathbf{U}_R)$ where $\mathbf{U}_L = (\rho_L, \rho_L u_L, E_L)^T$ and $\mathbf{U}_R = (\rho_R, \rho_R u_R, E_R)^T$. The speed of sound is given by $c^2 = \Psi + \Upsilon p/\rho$, where $\Psi = \partial p/\partial\rho|_e$ and $\Upsilon = (1/\rho)\,\partial p/\partial e|_\rho$. Throughout this work, a stiffened gas equation of state is used, given by $p = (\gamma - 1)\,\rho e - \gamma p_c$. For this equation of state, $\Psi = (\gamma - 1)\,e$ and $\Upsilon = \gamma - 1$. Following Roe (1986) and Glaister (1988) the averages $\tilde{\rho}$, $\tilde{u}$, $\tilde{\Psi}$ and $\tilde{\Upsilon}$ are calculated according to

$$\tilde{\rho} = \sqrt{\rho_L \rho_R}, \tag{A.12}$$

and

$$\tilde{f} = \mu(f) = \frac{\sqrt{\rho_L} f_L + \sqrt{\rho_R} f_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{A.13}$$

for $f = u$, Υ and Ψ. The average of p/ρ is given by

$$\left(\tilde{\frac{p}{\rho}}\right) = \mu\left(\frac{p}{\rho}\right) + \frac{1}{2}\left(\frac{u_R - u_L}{\sqrt{\rho_L} + \sqrt{\rho_R}}\right)^2. \tag{A.14}$$

The average sound speed is then calculated from

$$\tilde{c}^2 = \tilde{\Psi} + \tilde{\Upsilon}\left(\tilde{\frac{p}{\rho}}\right). \tag{A.15}$$

The left and right wave speeds are estimated according to

$$S_L = \min\left(u_L - c_L,\, \tilde{u} - \tilde{c}\right), \qquad S_R = \max\left(\tilde{u} + \tilde{c},\, u_R + c_R\right). \tag{A.16}$$

Then the contact wave velocity is given by

$$u^\star = S^\star = \frac{\rho_R u_R\left(S_R - u_R\right) + \rho_L u_L\left(u_L - S_L\right) + p_L - p_R}{\rho_R\left(S_R - u_R\right) + \rho_L\left(u_L - S_L\right)}, \tag{A.17}$$

and the pressure in the star states is

$$p^\star = p_L + \rho_L\left(u_L - S_L\right)\left(u_L - u^\star\right) = p_R + \rho_R\left(S_R - u_R\right)\left(u^\star - u_R\right). \tag{A.18}$$

Calculation of the energies in the star states is then straightforward, using the equation of state.

For the two-phase Riemann problems at the material interface, the star states are used directly to determine ghost cell states. For the single phase Riemann problems solved to calculate fluxes between cells, the star states and wave speeds are then used to obtain numerical fluxes, as described in the next section.

A.3 HLLC flux

At various times during the course of this work I have experimented with different finite volume schemes, and have implemented the schemes of Godunov (1959), Glimm (1965), Roe (1997), Liou and Steffen (1993) and Toro *et al.* (1994). Of these, I have found the HLLC scheme of Toro *et al.* (1994) performs most reliably, being very robust, and comparatively cheap and simple to implement. I use the HLLC flux in the scheme described in Chapter 4, with which all the results in Chapters 4 to 8 were generated.

The HLLC flux is defined as

$$\mathbf{F}_{HLLC} = \begin{cases} \mathbf{F}_L & 0 \leq S_L \\ \mathbf{F}_L + S_L \left(\mathbf{U}_L^\star - \mathbf{U}_L \right) & S_L \leq 0 \leq S^\star \\ \mathbf{F}_R + S_R \left(\mathbf{U}_R^\star - \mathbf{U}_R \right) & S^\star \leq 0 \leq S_R \\ \mathbf{F}_R & 0 \geq S_R, \end{cases} \tag{A.19}$$

where S_L, $S^\star$ and S_R are the three characteristic wave speeds from left to right, as obtained using the approximate Riemann solver of the previous section, and $\mathbf{F}_K = \mathbf{F}\left(\mathbf{U}_K\right)$ for $K = L$, R.

A.4 AUSM flux

Whilst the results within the body of the thesis use the HLLC flux, I use the AUSM flux to generate some of the results in Appendix B. The following is a description of the AUSM flux, and follows Liou and Steffen (1993). The flux $\mathbf{F}$ is defined

$$\mathbf{F} = \left[\begin{array}{ccc} \rho u, & \rho u^2 + p, & u\left(E + p\right) \end{array} \right]^T. \tag{A.20}$$

Let $\mathbf{F}$ be split such that $\mathbf{F} = \mathbf{F}^c + \mathbf{F}^p$, where the convective flux is defined as $\mathbf{F}^c = u \left[\begin{array}{ccc} \rho, & \rho u, & E + p \end{array} \right]^T$, and the pressure flux is $\mathbf{F}^p = \left[\begin{array}{ccc} 0, & p, & 0 \end{array} \right]^T$. I wish to find a numerical flux for the location $i + \frac{1}{2}$, $\hat{\mathbf{F}}_{i+\frac{1}{2}}$, between the i-th and the $(i+1)$-th cell, based on the solution in those two cells, $\mathbf{U}_i$ and $\mathbf{U}_{i+1}$. I denote $\mathbf{U}_L = \mathbf{U}_i$ and $\mathbf{U}_R = \mathbf{U}_{i+1}$. The convective term is given by

$$\hat{\mathbf{F}}^c_{i+\frac{1}{2}} = M_{i+\frac{1}{2}} \left[\begin{array}{ccc} \rho c, & \rho c u, & c\left(E + p\right) \end{array} \right]^T_{L/R}, \tag{A.21}$$

where c is the speed of sound, $M_{i+\frac{1}{2}}$ is an approximation to the advective velocity, and

$$\left(\cdot\right)_{L/R} = \begin{cases} \left(\cdot\right)_L & \text{if } M_{\imath+\frac{1}{2}} \geq 0, \\ \left(\cdot\right)_R & \text{otherwise.} \end{cases} \tag{A.22}$$

There are many ways in which $M_{i+\frac{1}{2}}$ may be calculated. Liou and Steffen (1993) use the following method. Set $M_{i+\frac{1}{2}} = M_L^+ + M_R^-$, and then define

$$M^{\pm} = \begin{cases} \pm\frac{1}{4}\left(M \pm 1\right)^2 & \text{if } |M| \leq 1, \\ \frac{1}{2}\left(M \pm |M|\right) & \text{otherwise.} \end{cases} \tag{A.23}$$

The pressure flux is defined as

$$\hat{\mathbf{F}}^p_{i+\frac{1}{2}} = \left[\begin{array}{ccc} 0, & p_{i+\frac{1}{2}}, & 0 \end{array} \right]^T. \tag{A.24}$$

Again there are many possible choices of $p_{i+\frac{1}{2}}$. Liou and Steffen (1993) set $p_{i+\frac{1}{2}} = p_L^+ + p_R^-$, and then define

$$p^{\pm} = \begin{cases} \frac{p}{2}\left(1 \pm M\right) & \text{if } |M| \leq 1, \\ \frac{p}{2}\left(M \pm |M|\right)/M & \text{otherwise.} \end{cases} \tag{A.25}$$

This completes the definition of the numerical flux vector.

Appendix B

Conservation errors in divergent coordinate systems

The Euler equations are a set of conservation laws for mass, momentum and energy which describe the motion of an inviscid fluid. They are often solved numerically using finite volume methods, as described in Chapter 2. In Chapter 4 I describe a finite volume scheme for their solution. The scheme uses a polar coordinate system, and a ghost fluid method to model an air-water interface. In Chapters 4, 5 and 6 I refer to a problem caused by the operator splitting method used to account for the geometry of the coordinate system. This problem appears to be inadequately discussed in the literature, and so in this appendix I describe this problem in more detail. In Section B.1 I give some background to the problem. In Section B.2 I describe the symptoms of this problem. In Section B.3 I mention a few relevant works in which this problem is, or ought to be, present. In section B.4 I discuss a potential solution to this problem. Section B.5 is a summary of conclusions.

B.1 The problem

The momentum component of the Euler equations contains a divergence term $\nabla \cdot (\mathbf{u} \otimes (\rho \mathbf{u}))$, where $\otimes$ is the tensor product, and a gradient term ∇p. In non-divergent coordinate systems, such as a Cartesian coordinate system, expressing the Euler equations in conservation form is straightforward, and the two terms are combined and written $\nabla \cdot (\mathbf{u} \otimes \rho \mathbf{u} + \mathbf{I} p)$, where $\mathbf{I}$ is the identity matrix. However, in divergent coordinate systems, such as polar or cylindrical coordinate systems, it is more complex. The usual method employed by finite volume schemes in divergent coordinate systems is to write the Euler equations in a non-conservative form, via the use of source terms, and then use an operator splitting procedure. There are several different ways in which the Euler equations may be split to include source terms. The two most common ones are: (1) treat both advective fluxes and pressure gradients as gradients in a Cartesian coordinate system, then correct for this with source terms in all three components of the Euler equations, or (2) treat both advective fluxes and pressure terms as divergent fluxes in the divergent coordinate system, then correct for the pressure gradient with a source term in the momentum equation. Both approaches are non-conservative. Regarding the issues discussed in this appendix, both approaches lead to the same conservation errors. Throughout this thesis I take approach (1), and so I focus on approach (1) in this appendix.

Following approach (1) the Euler equations are written

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial r} + \mathbf{S}(\mathbf{U}) = 0, \tag{B.1}$$

where the vector of conservative properties $\mathbf{U}$ is given by

$$\mathbf{U} = \left[\begin{array}{ccc} \rho, & \rho u, & E \end{array} \right]^T, \tag{B.2}$$

the vector of fluxes $\mathbf{F}$ is

$$\mathbf{F}(\mathbf{U}) = \left[\begin{array}{ccc} \rho u, & \rho u^2 + p, & u(E+p) \end{array} \right]^T, \tag{B.3}$$

and the vector of geometric source terms $\mathbf{S}$ due to the coordinate system is

$$\mathbf{S}(\mathbf{U}) = \frac{\alpha u}{r} \left[\begin{array}{ccc} \rho, & \rho u, & (E+p) \end{array} \right]^T. \tag{B.4}$$

The spatial coordinate is r, and t denotes time. ρ is the density, u the velocity, p the pressure and E the total energy. α describes the coordinate system used, with $\alpha = 0, 1, 2$ representing Cartesian, cylindrical and polar coordinates respectively. Clearly in Cartesian coordinates the geometric source terms are zero. Equation B.1 is closed with an equation of state of the form $p = p(\rho, e)$, where e is related to E by $E = \rho e + \frac{1}{2}\rho u^2$.

Consider a domain Ω defined by $0 \leq r \leq 1$, constructed of N cells, each with width δr. The position of the centre of the i-th cell is r_i. On this discretised domain the conservative properties in the i-th cell at the n-th time step are denoted $\mathbf{U}_i^n$. By discretising in space and time, and applying Gauss' theorem to equation B.1, the first-order finite volume equation is obtained

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\delta t}{\delta r}\left\{\hat{\mathbf{F}}^n_{i+\frac{1}{2}} - \hat{\mathbf{F}}^n_{i-\frac{1}{2}}\right\} - \delta t \mathbf{S}\left(\mathbf{U}_i^n\right). \tag{B.5}$$

In the present work I use piecewise constant reconstruction to determine cell face properties, then use a Riemann solver due to Hu *et al.* (2009) to obtain HLLC fluxes $\hat{\mathbf{F}}^n_{i+\frac{1}{2}}$ and $\hat{\mathbf{F}}^n_{i-\frac{1}{2}}$. Equation B.5 may be solved using a two-step operator splitting procedure given by:

$$\mathbf{U}_i^\star = \mathbf{U}_i^n - \frac{\delta t}{\delta r}\left\{\hat{\mathbf{F}}^n_{i+\frac{1}{2}} - \hat{\mathbf{F}}^n_{i-\frac{1}{2}}\right\}, \tag{B.6a}$$

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^\star - \delta t \mathbf{S}\left(\mathbf{U}_i^\star\right). \tag{B.6b}$$

The value of δt is set by the Courant condition such that

$$\delta t = \frac{\mathrm{CFL}\delta r}{\max\left(|u| + c\right)}, \tag{B.7}$$

in which c is the local speed of sound, and $0 < \mathrm{CFL} < 1$. Typically, I set $\mathrm{CFL} = 0.8$. This is essentially a one-dimensional version of the scheme described in Chapter 4, Section 4.3. For simplicity, in this appendix I consider there to be zero-flux boundary conditions applied at both ends of the computational domain.

B.2 The symptoms

Consider a numerical scheme which solves equation B.6a with perfect conservation properties. For non-zero $\mathbf{S}$, the two-step operator splitting procedure of equations B.6a and B.6b may be non-conservative. This is because in solving equation B.6b the conservation-satisfying solution $\mathbf{U}^\star$ is augmented by a certain amount in each cell, such that the sum of the augmentations over Ω need not be zero. This fundamental problem is due to the non-conservative form of equation B.1. In many cases - if velocities are small, or density jumps are small - these errors are negligible. If the solution is smooth, the errors are likely to go unnoticed.

The spherically symmetric underwater explosion problem is an initial value problem where a sphere of high pressure gas is released in water and allowed to expand. This problem was first modelled by Flores and Holt (1981). A benchmark numerical solution is given by Wardlaw and Mair (1998). In some works the gas is governed by a Jones-Wilkins-Lee equation of state (Dobratz and Crawford, 1985), but the results are qualitatively unaffected if an ideal gas equation of state is used to treat the gas as air. The gas forms a bubble which oscillates. The problem has a large range of time-scales, as the speed at which the gas-water interface moves is, for much of the oscillation period, more than an order of magnitude less than the speed of propagation of the pressure wave transmitted into the water. The underwater explosion problem is a challenge for multi-material simulations, due to the drastically different properties of the gas and water, and is frequently used as a test problem for numerical schemes.

The solution to an underwater explosion problem contains large radial velocities. There is a density jump at the interface, and the thermodynamic properties (the equation of state) of the two fluids are very different. At the air-water interface there is a significant discontinuity in all three elements of $\mathbf{S}$. When the operator splitting procedure is applied, the solution fails to adhere to the Rankine-Hugoniot conditions. This error is usually most visible in the form of a pressure discontinuity at the interface. The magnitude of the pressure discontinuity is proportional to $\alpha u \delta r / r$. Figure B.1 shows

the pressure discontinuity when the gas-water interface is as $R_{int} = 0.4$ for different values of δr. Note that the magnitude of the pressure jump is proportional to δr.

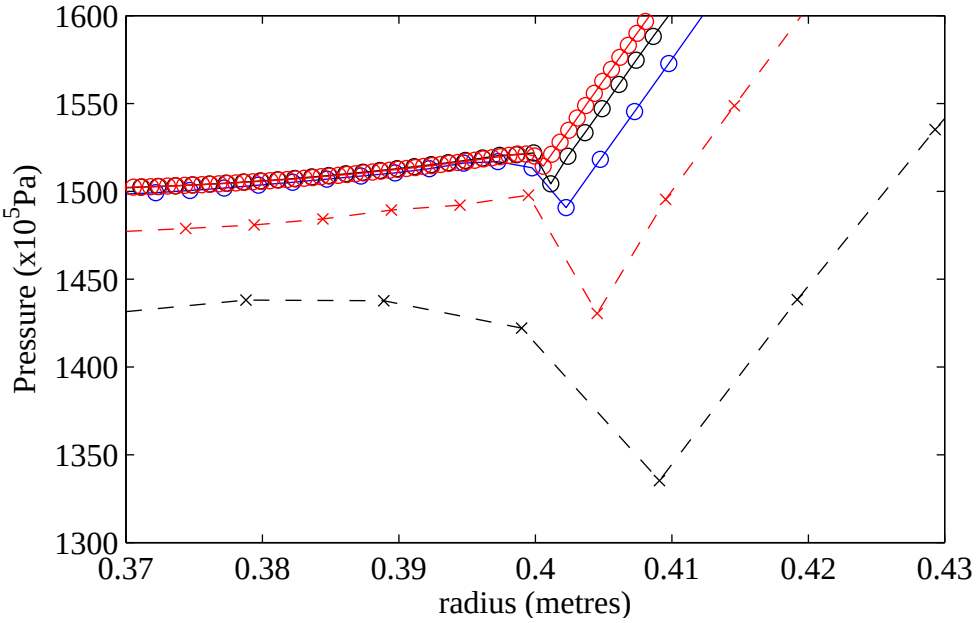


Figure B.1: The solution to the underwater explosion problem in the region around the interface when the gas-water interface reaches $R_{int} = 0.4$. The different traces correspond to different values of δr. Dashed black line with crosses - $\delta r = 0.01$; dashed red line with crosses - $\delta r = 0.005$; solid blue line with circles - $\delta r = 2.5 \times 10^{-3}$; solid black line with circles - $\delta r = 1.25 \times 10^{-3}$; solid red line with circles - $\delta r = 6.25 \times 10^{-4}$.

B.3 In the literature

Flores and Holt (1981), Cocchi *et al.* (1996), Wardlaw and Mair (1998), Liu *et al.* (2001a), Hu *et al.* (2006), Muller *et al.* (2009), Pishevar and Amirifar (2010) and Shaw and Spelt (2010) all simulate the underwater explosion problem using an operator splitting technique as described in the previous section, although in other respects the methods of solution are quite varied (for instance, through the use of different reconstruction schemes or numerical fluxes, and in some cases through the use of arbitrary Lagrangian-Eulerian (ALE) schemes). Cooke and Chen (1991) also use an

operator splitting technique, but the source terms they describe are incorrect. In all these works, a pressure discontinuity should be expected at the interface.

The pressure discontinuity is visible in only a few papers, for example Luo *et al.* (2004) and Shaw and Spelt (2010). The pressure discontinuity can be reduced by using a fine mesh, and the use of a very fine mesh for this type of problem should immediately arouse suspicion. Results obtained using a very fine mesh are presented by, for example, Luo *et al.* (2004) Hu *et al.* (2006) and Pishevar and Amirifar (2010). Some authors place a symbol over the interface to indicate its position, which has the side effect of obscuring the pressure profile close to the interface (Wardlaw and Mair (1998), Luo *et al.* (2004) and Pishevar and Amirifar (2010)). One work even places the word 'interface' (Liu *et al.*, 2001a) over the interface.

B.4 A potential solution

The 'advective upstream splitting method' (AUSM) family of fluxes (Liou and Steffen, 1993; Liou, 1996) is based on the splitting of fluxes into the convection and pressure fluxes. Convection fluxes move at the local velocity, whilst pressure fluxes advect at the sound speed. Full details of the AUSM flux are given in Appendix A. Importantly, this allows the curvature due of the coordinate system to be accounted for without the use of source terms. In general, the splitting process is only used in the calculation of the fluxes, and the components are then recombined prior to use in equation B.5. The pressure component of the flux may adjusted by some factor to account for the geometry. This method is employed by Smith (1999), in whose results the error is not visible, although the results are obtained with a very fine mesh.

A slight variation to this approach is to start from the Euler equations as written by Colella (1985)

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \left(A\mathbf{F} \right)}{\partial V} + \frac{\partial \mathbf{H}}{\partial r} = 0, \tag{B.8}$$

where $\mathbf{U}$ is as defined in the previous sections, $\mathbf{F} = \left[\begin{array}{ccc} \rho, & \rho u, & E \end{array} \right]^T$ and $\mathbf{H} =$

$\left[\ 0, p, 0\ \right]^T$. A is the volume coordinate, given by $V = r^{\alpha+1}/(\alpha+1)$ and $A = dV/dr$ is the area. The finite volume equation B.5 is then written

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\delta t}{\delta V}\left\{A_{i+\frac{1}{2}}\hat{\mathbf{F}}^n_{i+\frac{1}{2}} - A_{i-\frac{1}{2}}\hat{\mathbf{F}}^n_{i-\frac{1}{2}}\right\} - \frac{\delta t}{\delta r}\left\{\hat{\mathbf{H}}_{i+\frac{1}{2}} - \hat{\mathbf{H}}_{i-\frac{1}{2}}\right\}, \tag{B.9}$$

where $\hat{\mathbf{F}}$ is the numerical flux approximation to $\mathbf{F}$ - the convective component of the AUSM flux - and $\hat{\mathbf{H}}$ is the numerical flux approximation to $\mathbf{H}$ - the pressure component of the AUSM flux. This approach can in fact be used with a version of the HLLC flux, where the convective and pressure terms are separated. I refer to this method as the split HLLC flux.

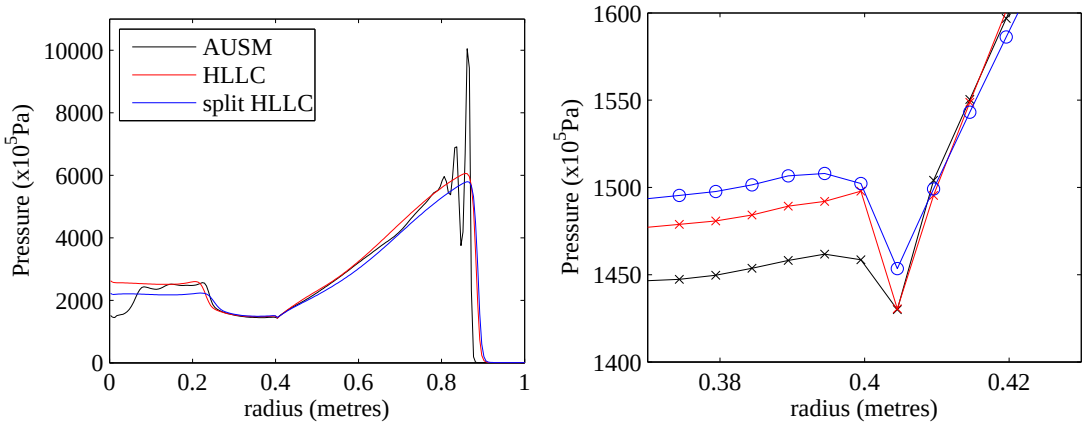


Figure B.2: The solution to the underwater explosion problem when the gas-water interface reaches $R_{int} = 0.4$, with different approaches to the spherical geometry. The left panel shows the solution over the entire domain. The right panel shows the region around the interface. The red trace shows the solution using the approach described in Section B.1 with the HLLC flux. The black traces corresponds to the method described above using the AUSM flux. The blue trace corresponds to the split-HLLC method.

Figure B.2 shows the solution to the underwater explosion problem when the interface reaches $R_{int} = 0.4$, using different approaches to the geometry. The solution obtained using the method described in Section B.1 (and used in the scheme described in Chapter 4) is shown by the red trace. Note that in the right panel, this trace shows the largest pressure discontinuity at the interface. The method of splitting described above is used with two fluxes, the AUSM (shown in black) and the HLLC (shown in blue). Unfortunately, the AUSM scheme is unstable around the outgoing shock, as can

clearly be seen in the left panel of Figure B.2. However, the right panel does show a reduction in the pressure discontinuity using the AUSM flux. The split-HLLC scheme shows a slight reduction in the magnitude of the pressure discontinuity compared with the unsplit scheme, though not to the same extent as with the AUSM flux. Importantly, the split-HLLC scheme shows slightly different behaviour from the unsplit scheme in other respects. These are visible in the left panel of Figure B.2, where the strengths of the shock and rarefaction waves are different. Whilst there is no analytic solution to the underwater explosion problem, there exist measurements and a widely accepted benchmark numerical solution (Wardlaw and Mair, 1998), and it is known that curves of bubble radius and bubble pressure against time should be similar to those in Figure 5.10 in Chapter 5. Unfortunately, the differences visible in the left panel of Figure B.2 - due to differences in the transmission of momentum and energy across the interface - lead to drastically different amounts of damping in bubble oscillations. The split-HLLC scheme shows almost no oscillatory behaviour in the long term, as the damping is too great. Whilst these approaches do reduce the pressure discontinuity at the interface, both the split-HLLC and AUSM schemes have other drawbacks, which mean they cannot provide an adequate solution to the problem.

There has been much work on flux splitting schemes. Whilst a thorough investigation on the effects of such schemes on spherical bubble oscillation simulations is beyond the scope of this work, I believe that this approach may provide a solution to the problem of pressure discontinuities at the interface.

B.5 Conclusions

During the course of this work I observed an error in the results of my code, whereby a pressure discontinuity sits on the material interface as the simulated bubble expands and contracts. Other authors have encountered this problem, although it is inadequately discussed in the literature. Extensive investigations led to an explanation of the cause of the problem. The non-conservative form of the Euler equations in polar coordinates

leads to conservation errors, most obvious around the interface. I do not present a complete solution to this problem, but I propose that flux-splitting schemes are an avenue of research down which a solution may lie.

Appendix C

Bubble surface stability analysis

C.1 Taylor's analysis

The interface between two fluids may be stable or unstable. If the interface is stable then small disturbances to the surface will be damped out, and it will tend to remain smooth. If the surface is unstable small disturbances to the surface will to grow over time. The instability which occurs when two fluids are accelerated in a direction perpendicular to the plane of the interface between them is referred to as Rayleigh-Taylor instability. Taylor (1950) derived criteria for the stability of a planar interface. This derivation may be applied to a spherical bubble, provided that the characteristic sizes of disturbances are small compared with the radius of the bubble. This allows the curvature of the interface to be neglected. Below I present the derivation of these stability criteria, following Taylor (1950).

I begin by making the following assumptions:

1. the wavelengths of disturbances are small compared with the radius of the bubble, and hence the interface may be approximated as planar;

2. the amplitides of disturbances are much smaller than the wavelengths of distur-

bances, hence the equations describing the motion due to disturbances may be linearised;

3. the fluids are inviscid and the flow is irrotational;

4. on the scales of the disturbances, the densities in both air and water can be considered uniform and constant;

5. all motion due to the disturbances has a negligible effect on the large scale motion of the bubble.

Consider a bubble with radius R. The bubble is expanding at a rate $\dot{R}$ and the acceleration of the bubble wall towards the water is $\ddot{R}$. Due to the first assumption above I consider a planar section of the interface. This is shown in Figure C.1. I define a coordinate system in a frame of reference moving with the interface, with the x-axis parallel with the plane of the surface and the y-axis perpendicular to the plane, positive into the bubble. Let the interface be subjected to a small sinusoidal disturbance η, with magnitude A and wavenumber k. The magnitude of A is small enough that terms of order A^2 may be neglected. The interface, along with the water and the contents of the bubble in the vicinity of the interface, are subject to an acceleration $\ddot{R}$, as shown. Properties within the bubble are denoted with the subscript A, and properties within the water W.

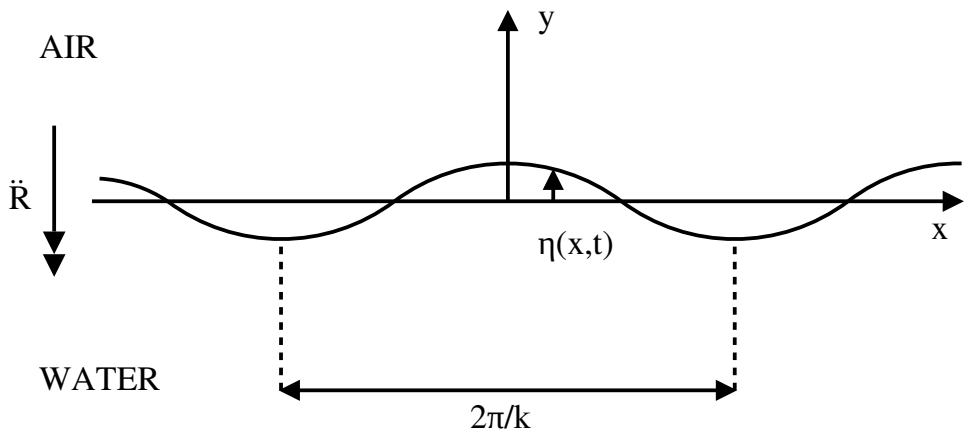


Figure C.1: Diagram of the disturbed air-water interface.

The shape of the disturbance is described by the equation

$$\eta(x,t) = Akn^{-1}e^{nt}\cos(kx). \tag{C.1}$$

Because the flow is irrotational and the motion *due to the disturbances* is incompressible, the motion due to the disturbances is described by a velocity potential ϕ, where the velocity is related to the velocity potential by $\mathbf{u} = \nabla\phi$, and the velocity potential obeys Laplace's equation

$$\nabla^2\phi = 0. \tag{C.2}$$

The velocities in the fluids at the interface must match the motion of the interface, hence

$$\left.\frac{\partial\phi}{\partial y}\right|_{y=0} = \frac{\partial\eta(x,t)}{\partial t}. \tag{C.3}$$

Using the method of separation of variables, writing $\phi(x,y,t) = f_{x,t}(x,t)\,f_y(y)$, it is straightforward to solve equation C.2 to find a general form of the velocity potential. Applying the boundary condition imposed by equation C.3, and the condition that the velocity must decay towards zero far from the interface, expressions for the velocity potential on both sides of the interface can be found

$$\phi_A = -Ae^{-ky+nt}\cos(kx) \tag{C.4a}$$

$$\phi_W = Ae^{ky+nt}\cos(kx). \tag{C.4b}$$

Bernoulli's equation for inviscid flow can be written as

$$p + \rho\frac{\partial\phi}{\partial t} + \frac{1}{2}\rho|\mathbf{u}|^2 - \rho\ddot{R}y = constant, \tag{C.5}$$

where ρ is the density and p is the pressure. From equations C.4 and the definition of ϕ, it follows that $|\mathbf{u}|^2 = \mathcal{O}\left(A^2\right)$, and hence the third term in equation C.5 can be neglected.

At the interface, where $y = \eta$,

$$\rho_A\frac{\partial\phi_A}{\partial t} - \rho_A\ddot{R}\eta = \rho_W\frac{\partial\phi_W}{\partial t} - \rho_W\ddot{R}\eta \tag{C.6}$$

Differentiation of equations C.4 gives

$$\frac{\partial\phi_A}{\partial t} = -Ane^{-ky+nt}\cos(kx) \tag{C.7a}$$

$$\frac{\partial\phi_W}{\partial t} = Ane^{ky+nt}\cos(kx). \tag{C.7b}$$

Substituting equations C.7 into equation C.6 yields

$$\rho_A A n e^{-k\eta+nt}\cos(kx) + \rho_A \ddot{R}\eta = -\rho_W A n e^{k\eta+nt}\cos(kx) + \rho_W \ddot{R}\eta. \tag{C.8}$$

$e^{k\eta}$ can be expressed as a power series according to

$$e^{k\eta} = 1 + k\eta + \mathcal{O}\left(\eta^2\right). \tag{C.9}$$

Neglecting terms of order A^2, equation C.8 becomes

$$\rho_A A n + \rho_A \ddot{R} A k n^{-1} = -\rho_W A n + \rho_W \ddot{R} A k n^{-1}, \tag{C.10}$$

which collapses to

$$\rho_A n^2 + \rho_A \ddot{R} k = -\rho_W n^2 + \rho_W \ddot{R} k. \tag{C.11}$$

By re-arranging equation C.11 an expression for n^2 is obtained

$$n^2 = \frac{\ddot{R}\left(\rho_W - \rho_A\right)k}{\left(\rho_W + \rho_A\right)} \tag{C.12}$$

When n is real and positive, the disturbances will grow in time, and the surface will be unstable. As k is always positive, and $\rho_W > \rho_A$, equation C.12 shows that the surface is unstable to *all* wavelengths of disturbance when $\ddot{R} > 0$, which occurs towards the end of collapse and at the start of expansion. At other times, the surface is stable.

C.2 The effect of surface tension

The Young-Laplace equation (Neumann *et al.*, 2010) relates the pressure difference across an interface to the surface tension and the curvature of the interface according to

$$\Delta p = -\sigma \nabla \cdot \mathbf{n}, \tag{C.13}$$

where $\mathbf{n}$ is a unit vector normal to the surface pointing towards the bubble, and σ is the surface tension. If the surface is expressed as the zero level of $F = y - \eta$, then $\mathbf{n} = \nabla F$, and $\mathbf{n}$ can be written

$$\mathbf{n} = \frac{A k^2 n^{-1} e^{nt}\sin(kx)\,\mathbf{i} + \mathbf{j}}{\left(1 + A^2 k^4 n^{-2} e^{2nt}\sin^2(kx)\right)^{\frac{1}{2}}}, \tag{C.14}$$

where **i** and **j** are the unit vectors aligned with the x- and y- axes respectively. Taking the divergence of C.14, substituting into equation C.13 and neglecting terms of order A^2 and higher gives the pressure difference across the interface

$$\Delta P = -\sigma A k^3 n^{-1} e^{nt} \cos(kx). \tag{C.15}$$

If the result of equation C.15 is added to the right hand side of equation C.6, and then the resulting expression is simplied as before, it can be shown that the range of wavenumbers for which the surface is unstable is

$$k^2 < \frac{\ddot{R}(\rho_W - \rho_A)}{\sigma}. \tag{C.16}$$

As the surface tension is increased, the upper limit of unstable wavenumbers is decreased, and hence the lower limit of stable wavelengths is decreased. Surface tension stabilises the interface to fine disturbances. The Young-Laplace equation is valid for static fluid interfaces. The interface in the present situation is not static, but there is no other simple expression available to relate the pressure jump across the interface to the surface tension and curvature. This may quantitatively affect the stability criteria, but the qualitative effect of surface tension in damping fine disturbances will remain.

Appendix D

Previous air gun bubble models

D.1 Introduction

The current state of the art in air gun modelling leaves much to be desired, a fact which is the main motivation for this work. In the main body of this thesis I develop a new model for air gun bubbles, but frequently refer to 'old models' or models based on homogeneous spherical bubbles. In this appendix I give some background to typical air gun bubble models, and I provide details of a typical air gun model of this type. The model I describe is not one used by any specific company. It is based on the model outlined by Ziolkowski and Metselaar (1984) recounted in several discussions with the first author, which is detailed in full in a confidential report by Ziolkowski (1986). It should be noted that whilst the physics in this model is similar to that in models used by industry, industry often manipulates these equations into a form which can be solved more efficiently.

The starting assumptions for most air gun bubble models are that the contents of the bubble are homogeneous, and that the bubble is spherical. This allows the bubble to be approximated by system of ordinary differential equations in the variables $R(t)$ and $P(t)$ where R is the radius of the bubble, P is the pressure within the bubble and t

is time. Typically, one equation relates R, P and their time-derivatives based on an approximation to the motion of the water. A polytropic equation of state for the bubble, or some set of equations representing the thermodynamics of the bubble, relates R and P to close the system.

The layout of this appendix is as follows. In Section D.2 I give some of the various equations describing the water. In Section D.3 I discuss some of the equations of state used for the bubble contents. In Section D.4 I detail the reconstructed model based on Ziolkowski and Metselaar (1984). Due to the large number of symbols introduced in this appendix which are not used elsewhere in the thesis, the symbols introduced in this appendix are not included in the nomenclature.

D.2 Equations for the motion in the water

Equations describing the motion of the water around a spherical oscillating bubble were developed by Rayleigh (1917), Herring (1941), Gilmore (1952), and Ziolkowski (1998), amongst others. Ziolkowski (1970) developed the first model for air gun bubble simulations, based on the equation of Gilmore (1952). This equation remains in use by industry today. Ziolkowski (1998) developed an improvement on the equation of Gilmore (1952), although this was first described by Herring (1941), albeit in a different form.

Rayleigh (1917) presents an equation based on the assumption that the water is incompressible. It is

$$\frac{P - p_\infty}{\rho} = R\ddot{R} + \frac{3\dot{R}^2}{2}, \tag{D.1}$$

where p_∞ is the pressure in the water at time zero, ρ is the density and $\dot{R}$ and $\ddot{R}$ are respectively the first and second time derivatives of R. The equation of Gilmore (1952) is

$$R\ddot{R}\left(1 - \frac{\dot{R}}{c}\right) + \frac{3\dot{R}^2}{2}\left(1 - \frac{\dot{R}}{3c}\right) = H\left(1 + \frac{\dot{R}}{c}\right) + \frac{R\dot{H}}{c}\left(1 - \frac{\dot{R}}{c}\right), \tag{D.2}$$

where c is the speed of sound and $H = \int \frac{dP}{\rho}$ is the enthalpy In this equation the water is assumed to be compressible with a constant speed of sound. For only small variations in density, which is the case for air gun bubble simulations, it is acceptable to use the approximation

$$H = \frac{P - p_\infty}{\rho_\infty}, \tag{D.3}$$

where ρ_∞ is the density of the undisturbed water. Ziolkowski (1998) developed an improved version of the equation due to Gilmore (1952), given by

$$R\ddot{R}\left(1 - \frac{2\dot{R}}{c}\right) + \frac{3\dot{R}^2}{2}\left(1 - \frac{4\dot{R}}{3c}\right) = H + \frac{R\dot{H}}{c}\left(1 - \frac{\dot{R}}{c}\right). \tag{D.4}$$

A major benefit of equation D.4 over equation D.2 is not apparent when the equations are in this form. The approximation on which equation D.4 is based allows a very simple calculation of the solution at any point in the water at time t, based on the values of R and P at times prior to t. Equation D.2 requires a more complex procedure and further approximations to extrapolate the solution at the bubble wall out into the far-field. A range of further equations of the same form was developed to describe cavitation bubbles, referred to as Rayleigh-Plesset equations (for example, Plesset (1949), Plesset and Zwick (1952), Prosperetti and Plesset (1978), Prosperetti and Lezzi (1986) and Lezzi and Prosperetti (1987)), details of which I omit.

D.3 Equations of state for the bubble

A range of equations of state for the bubble have been suggested. The simplest is the polytropic equation of state

$$PR^{3n} = \text{constant}, \tag{D.5}$$

where n is a constant. $n = 0$ would represent isobaric expansion, which is clearly unrealistic, as in this limit the bubble period would tend to infinity. $n = 1$ corresponds to an isothermal process. $n = \gamma$, where γ is the ratio of specific heats of the bubble contents, represents an isentropic process. $n = \infty$ is an isochoric process, and the bubble volume would remain constant. Ziolkowski (1970) found that $n = 1.13$ provided the

best fit of the bubble oscillation period with measured data. Ziolkowski (1970) argued that the value of n must lie between 1 and γ.

The bubble may be considered an open system. The rate at which work is done by the bubble on the water is $\dot{W}$. The rate at which heat is transferred from the water to the bubble is $\dot{Q}$. The kinetic energy of the air in the bubble can be neglected, and then the rate of change of temperature in the bubble $\dot{T}$ is given by

$$\dot{T} = \frac{\dot{Q} - \dot{W}}{m_b c_v}, \tag{D.6}$$

where c_v is the specific heat capacity at constant volume of the bubble contents, and m_b is the mass of the bubble. Equation D.6 is a statement of the first law of thermodynamics applied to the bubble. For an ideal gas

$$P = \rho R_b T \tag{D.7}$$

where R_b is the gas constant for the bubble contents and T is the temperature. From equations D.5 and D.7, if $n < 1$, as the bubble expands, the temperature in the bubble must increase. As the bubble expands, it does work on the water, and $\dot{W} > 0$. In order for the temperature to increase as the bubble expands, equation D.6 requires that $\dot{Q} > \dot{W} > 0$. The very low changes in density in the water imply that the temperature in the water is almost constant. Let the bubble and the water initially have the same temperature. As the bubble expands and heats, the second law of thermodynamics forbids the transfer of heat from the (colder) water, to the (hotter) bubble. Hence, $\dot{Q}$ cannot be positive, and $\dot{T}$ must also not be positive. A value of $n < 1$ violates the laws of thermodynamics. Hence the limit of $n \geq 1$ is valid, and represents the case of perfect heat transfer between the bubble and the water. Low values of n give reduced damping of oscillations, with a longer period. High values of n reduce the period and increase the damping. Because equation D.5 is used to encapsulate the entire behaviour of the bubble contents, which includes some fairly complex phenomena, there is no reason why the upper limit for n need be γ. Indeed, if the model described below, which is based on that of Ziolkowski and Metselaar (1984), were replaced with equation D.5, n would have to vary between $n = 1.4$ and $n = 3.5$ in order to obtain the same results.

More complex equations of state have been developed, which attempt to account for the

presence of water vapour, liquid water, and ice in the bubble, and also the effects of mass transfer by evaporation through the bubble surface. Whilst most of these models remain unpublished, Langhammer (1994) presents one such model as an appendix. Ziolkowski and Metselaar (1984) outline a model in which the contents of the bubble consist of air, water vapour, liquid water and ice. Based on Ziolkowski and Metselaar (1984) and private communication with the first author, I attempted to reconstruct this model.

D.4 The Ziolkowski-Metselaar model

Based on the short paper of Ziolkowski and Metselaar (1984), discussions with Anton Ziolkowski and a confidential technical report by Ziolkowski (1986), I attempted to reproduce an unpublished model developed in the 1980's by Anton Ziolkowski. In this section I present the details of this model. Ziolkowski and Metselaar (1984) claim the model includes the formation and melting of ice in the bubble. Whilst the Ziolkowski (1986) describes a system which contains ice, the figures in the report suggest there is no ice present. For the sake of simplicity, I neglect the presence of ice.

The motion in the water is described by equation D.2. An equation for the conservation of mass within the bubble is given by

$$m_v(t) + m_w(t) - m_v(0) - \int_0^t \dot{m}_{in} dt = 0, \tag{D.8}$$

where m_v is the mass of water vapour in the bubble, m_w is the mass if liquid water and $\dot{m}_{in}$ is the rate at which water vapour enters the bubble through evaporation. Equation D.8 states that the total mass of water in all forms in the bubble is equal to the initial mass of water in the bubble, plus the integral of the rate at which water has entered the bubble. The mass of air in the bubble m_a, is constant. An equation for the conservation of energy can be expressed as

$$\frac{d\left\{\Sigma_{i=a,v,w} m_i(t)\, c_{v,i} T(t)\right\}}{dt} + \dot{m}_{evap}(t)\, L_c + P(t)\, \dot{V}(t) = 0, \tag{D.9}$$

where $\dot{m}_{evap}$ is the rate at which liquid water evaporates to become water vapour in the bubble, L_c is the latent heat of condensation. P is the bubble pressure, T is the

bubble temperature and $\dot{V}$ is the rate of change of bubble volume. An equation of state for the gaseous contents of the bubble is given by

$$P(t)\,V(t) = \{m_a(t)\,R_{air} + m_v(t)\,R_{vap}\}\,T(t)\,, \tag{D.10}$$

where R_{air} and R_{vap} are the gas constants for air and water vapour respectively. The bubble is also constrained by the condition that the partial pressure of water vapour may not exceed the saturation vapour pressure. This is expressed by the relation

$$m_v(t) \leq \frac{\mathrm{SVP}\,(T(t))\,V(t)}{R_{vap}T(t)}, \tag{D.11}$$

where $\mathrm{SVP}\,(T)$ is the saturation vapour pressure at temperature T.

Equations D.8 to D.11, in conjunction with equation D.2 form the basis of the model. The volume of the bubble is related to the radius by

$$V(t) = \frac{4}{3}\pi R(t)^3\,. \tag{D.12}$$

The enthalpy in the water at the bubble wall is related to the pressure according to

$$H(t) = \frac{P(t) - p_\infty}{\rho_\infty}. \tag{D.13}$$

An empirical relation is used to calculate saturation vapour pressure from temperature, due to Wagner and Pruss (1993)

$$\ln\left\{\frac{\mathrm{SVP}\,(T)}{P_c}\right\} = \left[a_1\tau + a_2\tau^{1.5} + a_3\tau^3 + a_4\tau^{3.5} + a_5\tau^4 + a_6\tau^{7.5}\right]\frac{T}{T_c}, \tag{D.14}$$

where $\tau = 1 - T/T_c$ and T_c, P_c and a_1 to a_6 are constants. The rate of evaporation at the bubble surface is governed by the following equations

$$\dot{m}_{in}(t) = 4\pi R(t)^2\,F(t)\,\frac{1}{R_{vap}}\left[\frac{1}{T(t)}\frac{\partial P_v(t)}{\partial r}_{r=R} - \frac{P_v(t)}{T(t)^2}\frac{\partial T(t)}{\partial r}_{r=R}\right]D(t)\,, \tag{D.15}$$

where

$$\frac{\partial P_v(t)}{\partial r}_{r=R} = \frac{\mathrm{SVP}\,(T_{water}) - P_v(t)}{r_{diff}(t)}, \tag{D.16}$$

$$\frac{\partial T(t)}{\partial r}_{r=R} = \frac{T_{water} - T(t)}{r_{diff}(t)}, \tag{D.17}$$

$$D(t) = D_0\frac{P_0}{P(t)}\left[\frac{T(t)}{T_0}\right]^\alpha, \tag{D.18}$$

$$r_{diff}(t) = \frac{2D(t)}{\sqrt{2\pi\omega}}, \tag{D.19}$$

and

$$P_v(t) = P(t)\left[\frac{m_v(t)}{m_a + m_v(t)}\right], \tag{D.20}$$

in which T_0, P_0, D_0 and α are constants. F is the 'surface area factor', which may be a constant or a function of t, and is set to $F(t) = 624$ in order to achieve a good match with measured results.

Appendix E

Source code

If science is neither transparent nor repeatable, then its value is greatly diminished. I endeavour to ensure that my research is both transparent and repeatable. To this end, I provide the code developed during this work, from which all the results presented in this thesis may be obtained. During the course of this research I developed two codes - a one-dimensional finite volume code which was used to generate the results presented in Chapter 5, and a two-dimensional finite volume code, which was used to generate the results presented in Chapters 6, 7 and 8. This appendix contains a listing of these codes. Electronic copies of the codes are provided on the accompanying CD.

E.1 One-dimensional code

This section contains all the source code and parameter files required to generate the one-dimensional results shown in the thesis. The source code consists of 18 FORTRAN source files and a file containing a list of common variables. The source code is listed alphabetically according to filename, followed by four parameter files.

AUSM.f

```
1   SUBROUTINE AUSM(rhoL,momL,ENL,rhoR,momR,ENR,
    +           GAMR,PCRITR,
3   +           fl1,fl2,fl3)
```

```
      IMPLICIT NONE
C -------------------------------------------------------------
C SUBROUTINE TO CALCULATE AUSM FLUXES (Liou & Steffen 1993)
C -------------------------------------------------------------
C Returns the fluxes directly
C -------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   10<11-2014: CREATED
C -------------------------------------------------------------
      DOUBLE PRECISION  rhoL,momL,ENL,rhoR,momR,ENR,
     +        GAMR,PCRITR,rL,rR
C
      DOUBLE PRECISION  uL,PL,uR,PR,utL,utR
      DOUBLE PRECISION  eL,eR,HL,HR,cL,cR
      DOUBLE PRECISION  ML,MR,MLP,MRM,Mhalf
      DOUBLE PRECISION  pLp,pRm,phalf
C FLUXES!!
      DOUBLE PRECISION  fl1,fl2,fl3,fl4
C -------------------------------------------------------------
C CALCULATE THE OTHER VARIABLES FROM THE INPUTS...
      uL = momL/rhoL
      uR = momR/rhoR
      PL = (GAMR-1.0)*(ENL-0.5*rhoL*uL**2.0) - GAMR*PCRITR
      PR = (GAMR-1.0)*(ENR-0.5*rhoR*uR**2.0) - GAMR*PCRITR
c CALCULATE LEFT AND RIGHT SPEEDS OF SOUND ======================
      cL = sqrt(GAMR*(PL+PCRITR)/rhoL)
      cR = sqrt(GAMR*(PR+PCRITR)/rhoR)
C LITTLE ENERGY AND ENTHALPY
      eL = (PL+GAMR*PCRITR)/((GAMR-1.0)*rhoL)
      eR = (PR+GAMR*PCRITR)/((GAMR-1.0)*rhoR)
      HL = eL + PL/rhoL + 0.5*uL**2.0
      HR = eR + PR/rhoR + 0.5*uR**2.0
C The left and right Mach numbers and middle Mach number
      ML = uL/cL
      MR = uR/cR
      IF (ML.LE.1.0)THEN
      MLP = 0.25*(ML+1.0)**2.0
      MRM = -0.25*(MR-1.0)**2.0
      ELSE
      MLP = 0.5*(ML+abs(ML))
      MRM = 0.5*(MR-abs(MR))
      END IF
      Mhalf = MLP+MRM
C -------------------------------------------------------------
c middle pressure...
      IF (ML.LE.1.0)THEN
!     pLp = 0.25*pL*(2.0-ML)*(ML+1.0)**2.0 !alternative of L&S 93
!     pRm = 0.25*pR*(2.0+MR)*(MR-1.0)**2.0
      pLp = 0.5*pL*(1+ML)
      pRm = 0.5*pR*(1-MR)
      ELSE
      pLp = 0.5*pL*(ML+abs(ML))/ML
      pRm = 0.5*pR*(MR-abs(MR))/MR
      END IF
      phalf = pLp + pRm
C -------------------------------------------------------------
      IF (Mhalf.GT.0)THEN
        fl1 = Mhalf*rhoL*cL
        fl2 = Mhalf*rhoL*cL*uL + phalf
        fl3 = Mhalf*rhoL*cL*HL
      ELSE
        fl1 = Mhalf*rhoR*cR
        fl2 = Mhalf*rhoR*cR*uR + phalf
        fl3 = Mhalf*rhoR*cR*HR
```

```
  END IF
C ================================================================
C
  RETURN
  END
C ----------------------------------------------------------------
```

BOUNDLEFT.f

```
  SUBROUTINE BOUNDLEFT (mfl)

  INCLUDE "commonblock"
C ----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   20-07-2013: CREATED
C   06-08-2013: MODIFIED TO UPSTREAM WENO-Z SCHEME
C   30-09-2013: FOR A GENERAL NODE 'ibn' AT RIGHT BOUND
C   22-10-2013: BODGED. NB: in some cases, especially
C         when using 5th order WENO-Z and
C         3rd order TVD, variation at or-
C         -igin is such that 1-sided 3rd
C         order difference equation is
C         unstable. Hence the bodge.
C   11-02-2014: MODIFIED TO FIT NEW PROGRAM STRUCTURE
C ----------------------------------------------------------------
C THIS SUBROUTINE USES THE STANDARD CHARACTERISTIC BOUNDARY COND-
C ITION FORMULATION PRESENTED IN Thompson, JCP 89, 439-461(1990)
C TO APPLY A REFLECTIVE BOUNDARY AT THE LEFT EDGE OF THE DOMAIN
C ----------------------------------------------------------------
C ----------------------------------------------------------------
  INTEGER mfl
  DOUBLE PRECISION drhodrB,dudrB,dpdrB
  DOUBLE PRECISION cB,L1B,L2B,L3B
  DOUBLE PRECISION dudtB,drhodtB,dpdtB
  DOUBLE PRECISION newrho,newp,newu
C ----------------------------------------------------------------
C ----------------------------------------------------------------
C CALCULATE THE LOCAL SPEED OF SOUND ----------------------------
  cB = sqrt(gm(mfl)*(m0(mfl,4,rLOW(mfl))+pcr(mfl))/
     +    m0(mfl,1,rLOW(mfl)))
C ----------------------------------------------------------------
C FIRST CALCULATE DERIVATIVES OF THE PRIMITIVE VARIABLES AT THE -
C BOUNDARY: 1 SIDED 3rd ORDER -----------------------------------
  drhodrB = (11.0*m0(mfl,1,rLOW(mfl))- 18.0*m0(mfl,1,rLOW(mfl)+1)
     +      + 9.0*m0(mfl,1,rLOW(mfl)+2) -
     +      2.0*m0(mfl,1,rLOW(mfl)+3))/(-6.0*dr)
  dudrB = (11.0*m0(mfl,5,rLOW(mfl)) - 18.0*m0(mfl,5,rLOW(mfl)+1)
     +      + 9.0*m0(mfl,5,rLOW(mfl)+2) -
     +      2.0*m0(mfl,5,rLOW(mfl)+3))/(-6.0*dr)
  dpdrB = (11.0*m0(mfl,4,rLOW(mfl)) - 18.0*m0(mfl,4,rLOW(mfl)+1)
     +      + 9.0*m0(mfl,4,rLOW(mfl)+2) -
     +      2.0*m0(mfl,4,rLOW(mfl)+3))/(-6.0*dr)
C ----------------------------------------------------------------
C REFLECTIVE BOUNDARY! ------------------------------------------
  IF (m0(mfl,5,rLOW(mfl)) - cB .LT. 0.0) THEN
C   L1 OUTGOING - FROM DEFINITION
    L1B = (m0(mfl,5,rLOW(mfl))-cB)*(dpdrB -
     +      m0(mfl,1,rLOW(mfl))*cB*dudrB)
  ELSE
C   L1 INCOMING - SET TO ZERO (THIS SHOULD NEVER HAPPEN)
    L1B = 0.0
  END IF
  IF (m0(mfl,5,rLOW(mfl)) .LT. 0.0) THEN
C   L2 OUTGOING
```

```
58      L2B = m0(mfl,5,rLOW(mfl))*(cB*drhodrB - dpdrB)
        ELSE
60  C     L2 INCOMING
        L2B = 0.0
62      END IF
        IF (m0(mfl,5,rLOW(mfl)) + cB .LT. 0.0) THEN
64  C     L3 OUTGOING (THIS SHOULD NEVER HAPPEN)
        L3B = (m0(mfl,5,rLOW(mfl))+cB)*(dpdrB +
66       +       m0(mfl,1,rLOW(mfl))*cB*dudrB)
        ELSE
68  C     L3 INCOMING - SET TO L1B
        L3B = L1B
70      END IF
    C -------------------------------------------------------------
72  C CALCULATE TIME DERIVATIVES OF CHARACTERISTIC EULER EQUATIIONS -
    C USE abs(r(1)) RATHER THAN r(1) FOR SOME LONG FORGOTTEN REASON -
74      dudtB = (-0.5/(m0(mfl,1,rLOW(mfl))*cB))*(L3B-L1B)
        drhodtB = -1.0*((L2B + 0.5*(L3B+L1B))/cB**2
76         +       + coordsno*m0(mfl,1,rLOW(mfl))*
           +       m0(mfl,5,rLOW(mfl))/abs(r(rLOW(mfl)))
78         +       )
        dpdtB = -1.0*(0.5*(L3B+L1B)
80         +       + coordsno*m0(mfl,1,rLOW(mfl))*
           +       m0(mfl,5,rLOW(mfl))*cB*cB/abs(r(rLOW(mfl)))
82         +       )
    C -------------------------------------------------------------
84  C 1st ORDER TIME INTEGRATION ----------------------------------
        newrho = m0(mfl,1,rLOW(mfl)) + drhodtB*dt
86      newp = m0(mfl,4,rLOW(mfl)) + dpdtB*dt
        newu = m0(mfl,5,rLOW(mfl)) + dudtB*dt
88  C -------------------------------------------------------------
    C THIS IS A BODGE I USE SOMETIMES (all the time) ----------------
90      m0(mfl,1,rLOW(mfl)) = m0(mfl,1,rLOW(mfl)+1)
        m0(mfl,4,rLOW(mfl)) = m0(mfl,4,rLOW(mfl)+1)
92      m0(mfl,5,rLOW(mfl)) = -1.0*m0(mfl,5,rLOW(mfl)+1)
        m0(mfl,2,rLOW(mfl)) = m0(mfl,5,rLOW(mfl))*m0(mfl,1,rLOW(mfl))
94      m0(mfl,3,rLOW(mfl)) = m0(mfl,3,rLOW(mfl)+1)
    C -------------------------------------------------------------
96      RETURN
        END
98  C -------------------------------------------------------------
```

BOUNDRIGHT.f

```
        SUBROUTINE BOUNDRIGHT (mfl)
2
        INCLUDE "commonblock"
4   C -------------------------------------------------------------
    C AUTHOR: J. R. C. KING
6   C ---------------------
    C CHANGE RECORD:
8   C   20-07-2013: CREATED
    C   06-08-2013: MODIFIED TO UPSTREAM WENO-Z SCHEME
10  C   30-09-2013: FOR A GENERAL NODE 'ibn' AT RIGHT BOUND
    C   15-02-2014:   THOROUGHLY RE-SHUFFLED
12  C -------------------------------------------------------------
    C THIS SUBROUTINE USES THE STANDARD CHARACTERISTIC BOUNDARY COND-
14  C ITION FORMULATION PRESENTED IN Thompson, JCP 89, 439-461(1990)
    C TO APPLY BOUNDARY CONDITIONS AT THE RIGHT EDGE OF THE DOMAIN
16  C
    C USES SOLUTION AT RIGHT BOUNDARY CELL TO APPLY NLAA BC OR OTHER
18  C BC BY PROVIDING NEW VALUES FOR THIS CELL ----------------------
    C -------------------------------------------------------------
20  C -------------------------------------------------------------
        INTEGER mfl
22      DOUBLE PRECISION drhodrB,dudrB,dpdrB
```

```
      DOUBLE PRECISION cB,L1B,L2B,L3B
      DOUBLE PRECISION henth
      DOUBLE PRECISION aL1,NA,NB
      DOUBLE PRECISION dudtB,drhodtB,dpdtB
      DOUBLE PRECISION newrho,newp,newu
C -------------------------------------------------------------
C -------------------------------------------------------------
C CALCULATE THE LOCAL SPEED OF SOUND ----------------------------
      cB = sqrt(gm(mfl)*(m0(mfl,4,rHIGH(mfl))+pcr(mfl))/
     +    m0(mfl,1,rHIGH(mfl)))
C -------------------------------------------------------------
C FIRST CALCULATE DERIVATIVES OF PRIMITIVE VARIABLES AT THE -----
C BOUNDARY: 1 SIDED 3rd ORDER -----------------------------------
      drhodrB = (-11.0*m0(mfl,1,rHIGH(mfl)) +
     +      18.0*m0(mfl,1,rHIGH(mfl)-1) -
     +      9.0*m0(mfl,1,rHIGH(mfl)-2) +
     +      2.0*m0(mfl,1,rHIGH(mfl)-3))/(-6.0*dr)
      dudrB = (-11.0*m0(mfl,5,rHIGH(mfl)) +
     +      18.0*m0(mfl,5,rHIGH(mfl)-1) -
     +      9.0*m0(mfl,5,rHIGH(mfl)-2) +
     +      2.0*m0(mfl,5,rHIGH(mfl)-3))/(-6.0*dr)
      dpdrB = (-11.0*m0(mfl,4,rHIGH(mfl)) +
     +      18.0*m0(mfl,4,rHIGH(mfl)-1) -
     +      9.0*m0(mfl,4,rHIGH(mfl)-2) +
     +      2.0*m0(mfl,4,rHIGH(mfl)-3))/(-6.0*dr)
C -------------------------------------------------------------
C DETERMINE THE DIRECTIONS OF THE CHARACTERISTICS AND SET VALUES
C ACCORDINGLY ---------------------------------------------------
      IF (m0(mfl,5,rHIGH(mfl)) + cB .LE. 0.0) THEN
C   L3 INCOMING
        L3B = 0.0
      ELSE
C   L3 OUTGOING
        L3B = (m0(mfl,5,rHIGH(mfl))+cB)*(dpdrB +
     +      m0(mfl,1,rHIGH(mfl))*cB*dudrB)
      END IF
C -------------------------------------------------------------
C THE TYPE OF BOUNDARY CONDITION IS SET BY THE obtype FLAG ------
C IF WE'RE LOOKING AT FLUID 1 WE FORCE A REFLECTING (dudt=0)
C BOUNDARY.
      IF (m0(mfl,5,rHIGH(mfl))-cB .LT. 0.0) THEN
C   L1 INCOMING -----------------------------------------
        IF (obtype .EQ. 1 .AND. mfl .EQ. 2) THEN
C     NON-REFLECTING, AS IN Thompson II, 1993 -------
          L1B = 0.0
          dudtB = (-0.5/(m0(mfl,1,rHIGH(mfl))*cB))*
     +        (L3B-L1B)
        ELSE IF (obtype .EQ. 2 .OR. mfl .EQ. 1) THEN
C     CONSTANT VELOCITY!! ALSO COURTESY OF Thompson -
          L1B = L3B
          dudtB = (-0.5/(m0(mfl,1,rHIGH(mfl))*cB))*
     +        (L3B-L1B)
        ELSE IF (obtype .EQ. 3 .AND. mfl .EQ. 2) THEN
C     NLAA BCs!!!
C     CALCULATE THE RHS AND VELOCITY TIME DERIVATIVE
          henth = (m0(mfl,4,rHIGH(mfl))-Pinf)/
     +        m0(mfl,1,rHIGH(mfl))
          aL1 = (m0(mfl,1,rHIGH(mfl))*
     +        (m0(mfl,5,rHIGH(mfl))-cB)/
     +        r(rHIGH(mfl)))*((3-m0(mfl,5,rHIGH(mfl))
     +        /cB)*0.5*m0(mfl,5,rHIGH(mfl))**2 +
     +        2*m0(mfl,5,rHIGH(mfl))*cB -
     +        (1+m0(mfl,5,rHIGH(mfl))/cB)*henth)
          NA = 1.0 - 0.5*(m0(mfl,5,rHIGH(mfl))-cB)*
     +        m0(mfl,5,rHIGH(mfl))/cB**2
          NB = (1/(2*m0(mfl,1,rHIGH(mfl))*cB))*(L3B-aL1)
```

```
      dudtB = -1.0*NB/NA
      L1B = aL1 + m0(mfl,1,rHIGH(mfl))*
     +        (m0(mfl,5,rHIGH(mfl))-cB)*
     +        m0(mfl,5,rHIGH(mfl))*dudtB/cB
    END IF
  ELSE
C   L1 OUTGOING ------------------------------------------
    L1B = (m0(mfl,5,rHIGH(mfl))-cB)*(dpdrB -
     +      m0(mfl,1,rHIGH(mfl))*cB*dudrB)
    dudtB = (-0.5/(m0(mfl,1,rHIGH(mfl))*cB))*(L3B-L1B)
  END IF
  IF (m0(mfl,5,rHIGH(mfl)) .LT. 0.0) THEN
C   L2 INCOMING ------------------------------------------
    L2B = (m0(mfl,1,rHIGH(mfl))*m0(mfl,5,rHIGH(mfl))*
     +      m0(mfl,5,rHIGH(mfl))/r(rHIGH(mfl)))*(
     +      (henth/cB) - (2*m0(mfl,5,rHIGH(mfl))) +
     +      (0.5*m0(mfl,5,rHIGH(mfl))*
     +      m0(mfl,5,rHIGH(mfl))/cB)) +
     +      m0(mfl,1,rHIGH(mfl))*m0(mfl,5,rHIGH(mfl))*
     +      (1-m0(mfl,5,rHIGH(mfl))/cB)*dudtB
C   THIS IS THE ZERO ENTROPY FLUX CHEAT CAUSING MINIMAL  --
C   ERRORS... --------------------------------------------
    L2B = 0.0
  ELSE
C   L2 OUTGOING ------------------------------------------
    L2B = m0(mfl,5,rHIGH(mfl))*(cB*drhodrB - dpdrB)
  END IF
C ----------------------------------------------------------------
C DETERMINE THE TIME DERIVATIVES --------------------------------
  drhodtB = -1.0*((L2B + 0.5*(L3B+L1B))/cB**2
     +      + coordsno*m0(mfl,1,rHIGH(mfl))*
     +      m0(mfl,5,rHIGH(mfl))/r(rHIGH(mfl))
     +      )
  dpdtB = -1.0*(0.5*(L3B+L1B)
     +      + coordsno*m0(mfl,1,rHIGH(mfl))*
     +      m0(mfl,5,rHIGH(mfl))*cB*cB/r(rHIGH(mfl))
     +      )
C ----------------------------------------------------------------
C 1st ORDER TIME INTEGRATION ------------------------------------
  newrho = sfrac(1,kt)*m1(mfl,1,rHIGH(mfl)) + sfrac(2,kt)*
     +    m0(mfl,1,rHIGH(mfl)) + sfrac(3,kt)*drhodtB*dt
  newp = sfrac(1,kt)*m1(mfl,4,rHIGH(mfl)) + sfrac(2,kt)*
     +    m0(mfl,4,rHIGH(mfl)) + sfrac(3,kt)*dpdtB*dt
  newu = sfrac(1,kt)*m1(mfl,5,rHIGH(mfl)) + sfrac(2,kt)*
     +    m0(mfl,5,rHIGH(mfl)) + sfrac(3,kt)*dudtB*dt
C ----------------------------------------------------------------
  m0(mfl,1,rHIGH(mfl)) = newrho
  m0(mfl,4,rHIGH(mfl)) = newp
  m0(mfl,5,rHIGH(mfl)) = newu
  m0(mfl,2,rHIGH(mfl)) = newrho*newu
  m0(mfl,3,rHIGH(mfl)) = (newp+gm(mfl)*pcr(mfl))/(gm(mfl)-1) +
     +      0.5*newrho*newu**2
C ----------------------------------------------------------------
  RETURN
  END
C ----------------------------------------------------------------
```

commonblock

```
C ----------------------------------------------------------------
c THIS FILE DEFINES ALL COMMON VARIABLES.
  PARAMETER (nrmax = 100002, ntmax = 1000002, npmax = 10)
C
C GRID
  INTEGER nr,nrout
  DOUBLE PRECISION  Rd,r(nrmax),dr
```

```
8   COMMON  / grid /
       +    Rd,r,dr,nr,nrout
10  C FLAGS_ETC
    INTEGER outfreq,ofc1,soltype,inttype,ghost
12  COMMON  / flags_etc /
       +    outfreq,ofc1,soltype,inttype,ghost
14  C A MASK!
    LOGICAL    pmask(nrmax)
16  COMMON  / a_mask /
       +    pmask
18  C COORDINATES
    DOUBLE PRECISION coordsno
20  COMMON  / coordinates /
       +    coordsno
22  C TIME
    DOUBLE PRECISION  dt,t,CFL,sfrac(3,3)
24  INTEGER tn,nt,kt
    COMMON  / time /
26     +    dt,t,CFL,sfrac,tn,nt,kt
    C INITIAL FIELD
28  DOUBLE PRECISION  u0,rho0,rho1,gamma0,gamma1,Pc0,Pc1
    DOUBLE PRECISION  p0,p1
30  INTEGER intype
    COMMON  / initial_properties /
32     +    u0,rho0,rho1,gamma0,gamma1,Pc0,Pc1,p0,p1,intype
    C BOUNDARY
34  DOUBLE PRECISION  Pinf
    INTEGER     obtype
36  COMMON  / boundary /
       +    Pinf,obtype
38  C INTERFACE
    DOUBLE PRECISION  Rint,RintOLD,fPint
40  DOUBLE PRECISION  PI,UI,rhoLI,rhoRI,SL,SR
    INTEGER     q,qOLD
42  COMMON  / the_interface /
       +    Rint,RintOLD,fPint,PI,UI,rhoLI,rhoRI,SL,SR,q,qOLD
44  C PROPERTIES
    DOUBLE PRECISION  rho(nrmax),u(nrmax),P(nrmax),
46     +    gamm(nrmax),Pc(nrmax),E(nrmax),alpha(nrmax)
    COMMON  / flow_properties /
48     +    rho, u, P,gamm,Pc,E,alpha
    C DIFFERENT MEDIUM INC. GHOST STATES & ALSO MUSCL OUTPUT
50  DOUBLE PRECISION  m0(2,npmax,nrmax),m1(2,npmax,nrmax)
    DOUBLE PRECISION  gm(2),pcr(2)
52  INTEGER     gfmbuffer,rLOW(2),rHIGH(2)
    COMMON  / mediums /
54     +    m0,m1,gm,pcr,gfmbuffer,rLOW,rHIGH
    C FLUXES
56  DOUBLE PRECISION  fiph(3,nrmax)
    COMMON  / the_fluxes /
58     +    fiph
    C LEFT AND RIGHT STATES
60  DOUBLE PRECISION  Uiph(npmax,nrmax),Uimh(npmax,nrmax)
    COMMON  / left_right_states /
62     +    Uiph,Uimh
    C
64  PARAMETER (pie=3.14159)
    C ---------------------------------------------------------------
```

DUMPRESULTS.f

```
1   SUBROUTINE DUMPRESULTS

3   INCLUDE "commonblock"
    C ---------------------------------------------------------------
5   C AUTHOR: J. R. C. KING
```

```
C ---------------------
C CHANGE RECORD:
C   20-11-2012: CREATED
C   ??<08-2013: MODIFIED FOR FAR FIELD OUTPUT IF REQ
C -----------------------------------------------------------------
C SUBROUTINE TO WRITE TO OUTPUTS IF IT IS THE REQUIRED TIME
C -----------------------------------------------------------------
C -----------------------------------------------------------------
  INTEGER ofspc
C -----------------------------------------------------------------
C EITHER WRITE TO OUTPUT OR ADD 1 TO THE OUTPUT COUNTER ---------
  IF (ofc1 .GE. outfreq) THEN
C   IT IS TIME TO WRITE TO OUTPUT ------------------------
C   SET THE SPATIAL OUTPUTTING FREQUENCY -------------------
    ofspc = int(float(nr)/float(nrout))
C   WRITE TO SOME FILES!!
    DO k=1,int(float(nr)/float(ofspc))
      i=int(float(ofspc)*float(k))
      WRITE (7,*) r(i),rho(i)
      WRITE (8,*) r(i),p(i)
      WRITE (9,*) r(i),u(i)
      WRITE (10,*) r(i),gamm(i)
      WRITE (11,*) r(i),Pc(i)
      WRITE (12,*) r(i),E(i)
      WRITE (13,*) r(i),alpha(i)
    END DO
C   RESET THE OUTPUT COUNTER -----------------------------
    ofc1 = 1
  ELSE
C   IT ISN'T TIME TO WRITE TO OUTPUT SO ADD 1 TO THE ------
C   OUTPUT COUNTER ---------------------------------------
    ofc1 = ofc1 + 1
  END IF
C WRITE OTHER THINGS TO OUTPUT FILES..
  WRITE (20,*) t,dt
  WRITE (22,*) t,Rint
  WRITE (23,*) t,P(q)
  WRITE (24,*) t,p(nr)
  WRITE (25,*) t,u(nr)
  WRITE (26,*) t,rho(nr)
C -----------------------------------------------------------------
  RETURN
  END
C -----------------------------------------------------------------
```

EVOLVE.f

```
  SUBROUTINE EVOLVE (mfl)

  INCLUDE "commonblock"
C -----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   20-07-2013: CREATED
C   07-08-2013: SPHERICALITY BY SOURCE TERMS
C   07-08-2013: MODIFIED TO WORK ON CONSERVATIVE VARS
C   14-08-2013: STOPPED SOLVING DUPLICATE RIEMANN PROBS
C   14-08-2013: ONLY USE NECESSARY GHOST CELLS..
C   25-09-2013: 3rd ORDER TVD TIME INTEGRATION OPTION!
C -----------------------------------------------------------------
C EVOLVE PROPERTIES GIVEN FLUXES. SOURCE TERMS NOT TIME-SPLIT.
C -----------------------------------------------------------------
C -----------------------------------------------------------------
  INTEGER mfl
  DOUBLE PRECISION delV
```

```
      DOUBLE PRECISION aph,amh
      DOUBLE PRECISION wr,wu,wm,we,wp,S(3)
c
      DOUBLE PRECISION dpdr,pp,pm
      DOUBLE PRECISION sp1,sp2,sp3,sp4,sp5
C -----------------------------------------------------------------
C USE THE FLUXES TO UPDATE THE PROPERTIES ------------------------
      DO i=rLOW(mfl)+1,rHIGH(mfl)-1
C   ------------------------------------------------------
        delV = dr
        aph = 1.0
        amh = 1.0
        if(.false.)then !alternative method...
        delV = (1.0/(coordsno+1.0))*((r(i)+0.5*dr)**
     +    (coordsno+1.0)-(r(i)-0.5*dr)**(coordsno+1.0))
        aph = (r(i)+0.5*dr)**coordsno
        amh = (r(i)-0.5*dr)**coordsno
        end if
C   ------------------------------------------------------
C   HAVE NOT BOTHERED TO OPERATOR SPLIT... ----------------
C   ------------------------------------------------------
        S(1)=m0(mfl,2,i)
        S(2)=m0(mfl,2,i)*m0(mfl,2,i)/m0(mfl,1,i)
        S(3)=m0(mfl,2,i)*(m0(mfl,3,i)+m0(mfl,4,i))/m0(mfl,1,i)
        if(.false.)then ! alternative method...
        S(1)=0
        S(2)=0
        S(3)=0
        end if
        DO j=1,3
          m0(mfl,j,i) = sfrac(1,kt)*m1(mfl,j,i) +
     +      sfrac(2,kt)*m0(mfl,j,i) - sfrac(3,kt)*
     +      ((dt/delV)*(aph*fiph(j,i)-amh*fiph(j,i-1))
     +      +dt*coordsno*S(j)/r(i))
        END DO
        m0(mfl,4,i) = (gm(mfl)-1.0)*(m0(mfl,3,i)-0.5*
     +      m0(mfl,2,i)*m0(mfl,2,i)/m0(mfl,1,i)) -
     +      gm(mfl)*pcr(mfl)
        m0(mfl,5,i) = m0(mfl,2,i)/m0(mfl,1,i)
      END DO
C ================================================================
C -----------------------------------------------------------------
      RETURN
      END
C -----------------------------------------------------------------
```

FINDINTERFACE.f

```
      SUBROUTINE FINDINTERFACE

      INCLUDE "commonblock"
C -----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ----------------------
C CHANGE RECORD:
C   ??-05-2013: CREATED
C   06-08-2013: MODIFIED TO USE WENO-Z SCHEME DERIVS
C   21-11-2013: SET SPATIAL DERIV TO 1 FOR NOW
C -----------------------------------------------------------------
C SUBROUTINE TO FIND THE LOCATION OF THE ZERO LEVEL-SET
C -----------------------------------------------------------------
C -----------------------------------------------------------------
      DOUBLE PRECISION  av_alpha,dalphadr
      DOUBLE PRECISION  fp,fm
C -----------------------------------------------------------------
C FIND THE INDICES OF THE CELLS EITHER SIDE OF THE ZERO LEVEL SET
```

```
      DO i=1,nr
        IF (alpha(i)*alpha(i+1) .LE. 0.0) THEN
          IF (i .GT. q-4 .AND. i .LE. q+4) THEN
            q = i
          END IF
        END IF
      END DO
C CALCULATE SPATIAL DERIVATIVES WITH A 5th ORDER WENO-Z SCHEME --
C EXTRAPOLATE OUT NEAR THE BOUNDARIES. IT IS A BODGE, BUT AN ----
C ACCURATE ONE AT PRESENT. -------------------------------------
      IF (q .GE. 3 .AND. q .LE. nr-2) THEN
        dalphadr = (alpha(q-2) - 8*alpha(q-1) +
     +      8*alpha(q+1) - alpha(q+1))/(12.0*dr)
        call WENOZ(alpha(q-2),alpha(q-1),alpha(q),alpha(q+1),
     +      alpha(q+2),fm,fp)
        dalphadr = (fp-fm)/dr
      ELSE IF (q .LT. 3) THEN
        dalphadr = (alpha(q+1) - alpha(q))/dr
      ELSE
        dalphadr = (alpha(q) - alpha(q-1))/dr
      END IF
C NB: In the 1D case the level set is advected with a uniform
C velocity field, so the spatial derivative is constant in time.
C It is initially set to 1, and so will remain equal to 1. This
C bodge has virtually no effect on results, but MIGHT (if I
C remember correctly) help with analysis of conservation
C properties of the rGFM...
      dalphadr = 1.0  !this is a big cheat, but changes nothing in
c     in this instance
C INTERPOLATE SOMEHOW TO FIND Rint ------------------------------
C NB: Need to stick a more accurate interpolation here sometime
      Rint = r(q) - alpha(q)/
     +      max(dalphadr,-1.0*alpha(q)/dr)
C ---------------------------------------------------------------
C JUST IN CASE IT LIES EXACTLY ON A NODE!! (SEEMS UNLIKELY) -----
      DO i=1,nr
        IF (alpha(i) .EQ. 0.0) THEN
          WRITE (4,*) "CRIKEY!! INTERFACE ON NODE ",i,"!"
          q = i
          Rint = r(q)
        END IF
      END DO
C ---------------------------------------------------------------
      RETURN
      END
C ---------------------------------------------------------------
```

FLUXES.f

```
      SUBROUTINE FLUXES (mfl)

      INCLUDE "commonblock"
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   11-02-2014: CREATED
C ---------------------------------------------------------------
C CALCULATES FLUXES FROM CELL EDGE VALUES
C ---------------------------------------------------------------
C ---------------------------------------------------------------
      INTEGER mfl
C ---------------------------------------------------------------
C SOLVE THE RIEMANN PROBLEMS TO FIND THE FLUXES -----------------
      DO i=rLOW(mfl),rHIGH(mfl)-1
C   CALCULATE AT RIGHT "CELL WALLS": i,i+1
```

```
C   LEFT DENSITY, MOMENTUM, ENERGY,
C   RIGHT DENSITY, MOMENTUM, ENERGY,
C   FLUID PROPERTIES (gamma and Pc),
C   FLUXES OF DENSITY, MOMENTUM, ENERGY

!!!!!   Note this little switch which lets me use AUSM when
!!!!!   if feel like it! Note AUSM unstable for big shocks
    IF (.false.)THEN
      call AUSM
     +      (Uiph(1,i),Uiph(2,i),
     +      Uiph(3,i),
     +      Uimh(1,i+1),Uimh(2,i+1),Uimh(3,i+1),
     +      gm(mfl),pcr(mfl),
     +      fiph(1,i),fiph(2,i),fiph(3,i))
    ELSE
      call RIEMANNHLLC
     +      (Uiph(1,i),Uiph(2,i),
     +      Uiph(3,i),
     +      Uimh(1,i+1),Uimh(2,i+1),Uimh(3,i+1),
     +      gm(mfl),pcr(mfl),
     +      fiph(1,i),fiph(2,i),fiph(3,i))
    END IF
  END DO
C -----------------------------------------------------------------
  RETURN
  END
C -----------------------------------------------------------------
```

KINGEULER1D.f

```
  PROGRAM KINGEULER1D

  INCLUDE "commonblock"
C -----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C -----------------------------------------------------------------
  INTEGER spareINT
  DOUBLE PRECISION store1
C -----------------------------------------------------------------
C OPEN FILES ----------------------------------------------------
  OPEN(unit=1,file="init.params",status='old')
  OPEN(unit=7,file="rho.out")
  OPEN(unit=8,file="P.out")
  OPEN(unit=9,file="u.out")
  OPEN(unit=10,file="gamma.out")
  OPEN(unit=11,file="Pc.out")
  OPEN(unit=12,file="E.out")
  OPEN(unit=13,file="A.out")
  OPEN(unit=20,file="dt.out")
  OPEN(unit=22,file="interface.out")
  OPEN(unit=23,file="pint.out")
  OPEN(unit=24,file="pbound.out")
  OPEN(unit=25,file="ubound.out")
  OPEN(unit=26,file="rhobound.out")
C -----------------------------------------------------------------
C GET EVERYTHING READY FOR THE TIME LOOP ------------------------
  call SETUP
C SET TIME INDEX TO 1
  tn=1
  call FINDINTERFACE
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C THIS IS THE TIME LOOP -----------------------------------------
  DO tn = 1,nt
! DO WHILE (t.LE.0.0044) !alternative if want sod shock tube
!   tn=tn+1        ! ditto
```

```
C   ------------------------------------------------------
C   PROGRESS COUNTER EVERY NOW AND THEN -------------------
C   THIS IS WHAT PRINTS TO TERMINAL. IF IT FLICKERS -------
C   INCREASE VALUE OF spareINT. ---------------------------
    spareINT=200
    IF (mod(tn,spareINT).EQ.0) THEN
    call PROGRESS
    END IF
C   SET THE TIME STEP AND WRITE IT TO A FILE SOMEWHERE ----
    call SETTSTEP
C   WRITE RESULTS -----------------------------------------
    call DUMPRESULTS
C   ------------------------------------------------------
C   SET UP STATES FOR EACH MEDIUM ACCORDING TO GFM --------
C   Note that the rGFM is currently used once per complete
C   time step, rather than every sub-step of the TVD time
C   integration scheme. I cannot decide whether this is the
C   correct way to do things. Perhaps not, as if I use a
C   multi-step integration scheme it all goes wrong...
    call SPLITFIELDS
C   ------------------------------------------------------
C   UPDATE THE LEVEL SET ----------------------------------
    call LEVELSET
C   ------------------------------------------------------
C   FIND THE INTERFACE ------------------------------------
    call FINDINTERFACE
C   AN EULER SOLVER FOR EACH FLUID ------------------------
    DO k=1,2
C     PROPERTIES AT START OF TIME STEP --------------
      DO i=rLOW(k),rHIGH(k)
        DO j=1,5
          m1(k,j,i) = m0(k,j,i)
        END DO
      END DO
C     MULTI-STEP TIME INTEGRATION -------------------
      DO kt =1,inttype
C       BOUNDARY CONDITIONS AT LEFT OF FLUID --
        call BOUNDLEFT(k)
C       DOUNDARY CONDITIONS AT RIGHT OF FLUID -
        call BOUNDRIGHT(k)
C       RECONSTRUCTION FOR CELL EDGE PROPERTIES
        call RECONSTRUCT(k)
C       FIND THE FLUXES -----------------------
        call FLUXES(k)
C       EVOLVE!!! -----------------------------
        call EVOLVE(k)
      END DO
    END DO
C   ------------------------------------------------------
C   RECOMBINE FIELDS FROM GHOST FLUID METHOD --------------
    call RECOMBINE
C   UPDATE TIME -------------------------------------------
    t = t + dt
C   ------------------------------------------------------
  END DO
C THAT'S THE END OF THE TIME LOOP -----------------------------
  ! this is for getting sod results...
  OPEN(unit=56,file='final_rho')
  OPEN(unit=57,file='final_u')
  OPEN(unit=58,file='final_p')
  OPEN(unit=59,file='final_E')
  DO i=1,nr
    write(56,*)r(i),rho(i)
    write(57,*)r(i),u(i)
    write(58,*)r(i),p(i)
    write(59,*)r(i),E(i)
```

```
  END DO
104 do i=56,59
    close(i)
106 end do
C AND THAT'S THE END OF THE PROGRAM. STOP. END. GOODBYE! --------
108 C ---------------------------------------------------------------
  STOP
110 END
C ---------------------------------------------------------------
```

LEVELSET.f

```
1   SUBROUTINE LEVELSET

3   INCLUDE "commonblock"
C ---------------------------------------------------------------
5 C AUTHOR: J. R. C. KING
C ---------------------
7 C CHANGE RECORD:
C   20-07-2013: CREATED
9 C   06-08-2013: MODIFIED TO UPSTREAM WENO-Z SCHEME
C   21-11-2013: TIME INTEGRATION SAME ORDER AS EULER
11 C ---------------------------------------------------------------
C THIS SUBROUTINE UPDATES A LEVEL SET ACCORDING TO:
13 C da/dt + uLS*da/dr = 0
C A 5th ORDER UPSTREAM WENO-Z SCHEME IS USED FOR SPATIAL DISCRET-
15 C ISATION. TIME INTEGRATION IS OF THE ORDER OF THE TIME INTEGRAT-
C ION OF THE EULER SOLUTION (set by inttype).
17 C ---------------------------------------------------------------
C ---------------------------------------------------------------
19  DOUBLE PRECISION  dadr(nrmax)
  DOUBLE PRECISION  store1,spare1,aUPD(nrmax)
21  DOUBLE PRECISION  uls
C ---------------------------------------------------------------
23 C DECIDE THE LEVEL SET VELOCITY! -------------------------------
C OPTION 1 - the ghost cell velocity
25  uls = uI
C ---------------------------------------------------------------
27 C SET THE VALUES ------------------------------------------------
  DO i=1,nr
29    aUPD(i) = alpha(i)
  END DO
31 C ---------------------------------------------------------------
C ORDER OF INTEGRATION LOOP ----------------------------------------
33  DO k=1,inttype
C   ------------------------------------------------------
35 C   USE WENOZ TO CALCULATE DERIVATIVE OF ALPHA ------------
    IF (uI .GT. 0.0) THEN
37      DO i=4,nr-2
C         LEFT SIDED DERIVATIVES -----------------
39        call WENOZ ( (aUPD(i-2)-aUPD(i-3)),
     +         (aUPD(i-1)-aUPD(i-2)),
41   +         (aUPD(i)-aUPD(i-1)),(aUPD(i+1)-aUPD(i))
     +         ,(aUPD(i+2)-aUPD(i+1)),spare1,store1)
43        dadr(i) = store1/dr
      END DO
45    ELSE
      DO i=3,nr-3
47 C       RIGHT SIDED DERIVATIVES ---------------
        call WENOZ ((aUPD(i+3)-aUPD(i+2)),
49   +         (aUPD(i+2)-aUPD(i+1)),
     +         (aUPD(i+1)-aUPD(i)),(aUPD(i)-aUPD(i-1))
51   +         ,(aUPD(i-1)-aUPD(i-2)),spare1,store1)
        dadr(i) = store1/dr
53    END DO
    END IF
```

```
C   -------------------------------------------------------
C   BODGE BOUNDARY CONDITIONS(ACTUALLY EXACT FOR THIS CASE)
    DO i=1,3
      dadr(i) = dadr(4)
      dadr(i) = dadr(4)
    END DO
    DO i=nr-2,nr
      dadr(i) = dadr(nr-3)
      dadr(i) = dadr(nr-3)
    END DO
C   UPDATE THE LEVEL SET ---------------------------------
    DO i=1,nr
      aUPD(i)=sfrac(1,k)*alpha(i)+sfrac(2,k)*
     +      aUPD(i) - sfrac(3,k)*dt*uls*dadr(i)
    END DO
  END DO
C -----------------------------------------------------------------
C RETURN NEW VALUES TO ALPHA -------------------------------------
  DO i=1,nr
    alpha(i) = aUPD(i)
  END DO
C -----------------------------------------------------------------
  RETURN
  END
C -----------------------------------------------------------------
```

PROGRESS.f

```
  SUBROUTINE PROGRESS

  INCLUDE "commonblock"
C -----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   08-01-2014: CREATED
C -----------------------------------------------------------------
C SUBROUTINE TO PRINT SOME THINGS TO SCREEN. HOPEFULLY THIS LOOKS
C NICE ON A 24 LINE TERMINAL...
C -----------------------------------------------------------------
  WRITE (6,'(A24,I2)') 'KE1D running test case: ',intype
  WRITE (6,'(A43)') "==========================================="
  WRITE (6,'(A14,I3,A5,I5,A6)') 'Domain size = ',int(Rd),' with',
     +      nr,' cells'
  WRITE (6,'(A35,I1)') 'Spatial discretisation is of order ',
     +      soltype
  WRITE (6,'(A29,I1)') 'Time integration is of order ',inttype
  WRITE (6,'(A26,F3.1)') 'Coordinate system is type ',coordsno
  WRITE (6,'(A43)') "==========================================="
  WRITE (6,9) 'MAX P = ',maxval(p,pmask),'; MIN P = ',
     +        minval(p,pmask)
  WRITE (6,9) 'MAX u = ',maxval(u,pmask),'; MIN u = ',
     +        minval(u,pmask)
  WRITE (6,9) 'MAX rho = ',maxval(rho,pmask),
     +      '; MIN rho = ',minval(rho,pmask)
  WRITE (6,'(A43)') "==========================================="
  WRITE (6,'(A21,F5.2)') 'Interface location = ',Rint
  WRITE (6,'(A21,E8.2)') 'Interface pressure = ',p(q)
  WRITE (6,'(A43)') "==========================================="
  WRITE (6,'(A7,E10.4,A7,E10.4)') 'Time = ',t,'; dt = ',dt
  WRITE (6,'(A10,I6,A10,F8.4,A1)') 'time step:',tn,' Progress:',
     +      100.0*float(tn)/float(nt),'%'
  WRITE (6,10) ' '
  9 FORMAT(A8,E10.2,A10,E10.2)
  10  FORMAT(/,/,/,/,/,A1)
C -----------------------------------------------------------------
```

```
  RETURN
  END
C -----------------------------------------------------------------
```

RECOMBINE.f

```
  SUBROUTINE RECOMBINE

  INCLUDE "commonblock"
C -----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   11-02-2014: CREATED
C -----------------------------------------------------------------
C RECOMBINES m0(.,.,.) ACCORDING TO LEVEL SET VALUE...
C -----------------------------------------------------------------
  DO i=1,q
    rho(i) = m0(1,1,i)
    u(i) = m0(1,2,i)/m0(1,1,i)
    E(i) = m0(1,3,i)
    P(i) = m0(1,4,i)
    gamm(i) = gamma0
    Pc(i) = Pc0
  END DO
  DO i=q+1,nr
    rho(i) = m0(2,1,i)
    u(i) = m0(2,2,i)/m0(2,1,i)
    E(i) = m0(2,3,i)
    P(i) = m0(2,4,i)
    gamm(i) = gamma1
    Pc(i) = Pc1
  END DO
C -----------------------------------------------------------------
C -----------------------------------------------------------------
  RETURN
  END
C -----------------------------------------------------------------
```

RECONSTRUCT.f

```
  SUBROUTINE RECONSTRUCT (mfl)

  INCLUDE "commonblock"
C -----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   11-02-2014: CREATED
C -----------------------------------------------------------------
C RECONSTRUCTION SUBROUTINE. GIVEN FIELDS m0(mfl,.,.),
C RECONSTRUCTS TO FIND Uiph(.,.) AND Uimh(.,.). CHOICE OF PIECE-
C WISE CONSTANT, LINEAR OR WENO-Z RECONSTRUCTION, DETERMINED BY
C VALUE OF soltype
C -----------------------------------------------------------------
C -----------------------------------------------------------------
  INTEGER mfl,sltype
  DOUBLE PRECISION delU
C -----------------------------------------------------------------
C PIECEWISE CONSTANT RECONSTRUCTION ----------------------------
  DO i=rLOW(mfl),rHIGH(mfl)
    DO j=1,5
      Uiph(j,i) = m0(mfl,j,i)
      Uimh(j,i) = m0(mfl,j,i)
    END DO
```

```
      END DO
C -------------------------------------------------------------
C OR FOR MUSCLE SCHEME IF REQUESTED! ----------------------------
C PIECEWISE LINEAR RECONSTRUCTION OF DENSITY, VELOCITY, ---------
C RESSURE AND ENERGY. MOMENTUM CALCULATED FROM DENSITY ----------
C AND VELOCITY --------------------------------------------------
      IF (soltype .GT. 1) THEN
        DO i=rLOW(mfl)+1,rHIGH(mfl)-1
C   SNEAKY REDUCTION TO 1ST ORDER BESIDE INTERFACE --------
C         IF(mfl.EQ.1.AND.i.GE.q-1)THEN
C           sltype = 0
C         ELSE IF(mfl.EQ.2.AND.i.LE.q+2) THEN
C           sltype =0
C         ELSE
C           sltype = 1
C         END IF
C         OR I CAN JUST USE THE MUSCL SCHEME IN THE
C         BUBBLE AND NOT IN THE WATER!!
          IF (mfl.EQ.1.AND.i.LT.q-2)THEN
            sltype = 1
          ELSE
            sltype = 0
          END IF
          DO j=1,5
            call SLOPELIMITER (sltype,
     +         (m0(mfl,j,i)-m0(mfl,j,i-1))/
     +         (m0(mfl,j,i+1)-m0(mfl,j,i)),delU)
            Uiph(j,i) = m0(mfl,j,i) +
     +         delU*(m0(mfl,j,i+1)-m0(mfl,j,i))/2
            Uimh(j,i) = m0(mfl,j,i) -
     +         delU*(m0(mfl,j,i+1)-m0(mfl,j,i))/2
          END DO
          Uiph(2,i) = Uiph(1,i)*Uiph(5,i)
          Uimh(2,i) = Uimh(1,i)*Uimh(5,i)
        END DO
      END IF
C -------------------------------------------------------------
C -------------------------------------------------------------
C WHY NOT TRY A WENO-Z RECONSTRUCTION OF THE PRIMITIVE ----------
C VARIABLES? ----------------------------------------------------
      IF (soltype .EQ. 5) THEN
        DO i=rLOW(mfl)+2,rHIGH(mfl)-2
          DO j=1,5
            call WENOZ(m0(mfl,j,i-2),
     +         m0(mfl,j,i-1),m0(mfl,j,i),
     +         m0(mfl,j,i+1),m0(mfl,j,i+2),
     +         Uimh(j,i),Uiph(j,i))
          END DO
C         MOMENTUM AND ENERGY SET FROM PRIMITIVE --------
C         VARIABLES -----------------------------------
          Uiph(2,i) = Uiph(1,i)*Uiph(5,i)
          Uimh(2,i) = Uimh(1,i)*Uimh(5,i)
          Uiph(3,i) = (Uiph(4,i)+gm(mfl)*
     +       pcr(mfl))/(gm(mfl)-1.0)+0.5*Uiph(1,i)*
     +       Uiph(5,i)**2.0
          Uimh(3,i) = (Uimh(4,i)+gm(mfl)*
     +       pcr(mfl))/(gm(mfl)-1.0)+0.5*Uimh(1,i)*
     +       Uimh(5,i)**2.0
        END DO
      END IF
C =============================================================
C -------------------------------------------------------------
      RETURN
      END
C -------------------------------------------------------------
```

RIEMANNHLLC.f

```
1   SUBROUTINE RIEMANNHLLC (rhoL,momL,ENL,rhoR,momR,ENR,
      +          GAMR,PCRITR,
3     +          fl1,fl2,fl3)

5   IMPLICIT NONE
C -------------------------------------------------------------
7 C SUBROUTINE TO SOLVE A RIEMANN PROBLEM - HLLC SOLVER
C -------------------------------------------------------------
9 C DIRECTLY RETURNS FLUXES. TAKEN FROM
c http://dx.doi.org/10.1016/j.jcp.2009.06.002 (sort of)
11 C -------------------------------------------------------------
C AUTHOR: J. R. C. KING
13 C ---------------------
C CHANGE RECORD:
15 C   19-03-2013: CREATED
C   28-05-2013: MODIFIED TO RETURN FLUXES
17 C   29-07-2013: MODIFIED TO DEAL WITH ONLY 1 FLUID
C   07-08-2013: MODIFIED TO TAKE CONS VARS FOR INPUT
19 C -------------------------------------------------------------
  DOUBLE PRECISION  rhoL,momL,ENL,rhoR,momR,ENR,
21    +        GAMR,PCRITR
C
23  DOUBLE PRECISION  uL,PL,uR,PR
  DOUBLE PRECISION  pstar,ustar,rhoLstar,rhoRstar
25  DOUBLE PRECISION  ENLstar,ENRstar,eL,eR,HL,HR
  DOUBLE PRECISION  cL,cR
27  DOUBLE PRECISION  bL,bR,bM
  DOUBLE PRECISION  ubar,cbar,rhobar,Povrhobar,Pbar
29  DOUBLE PRECISION  hbar
  DOUBLE PRECISION  alf1,alf2,alf3
31 C FLUXES!!
  DOUBLE PRECISION  fl1,fl2,fl3
33  LOGICAL     subsonicflag
C -------------------------------------------------------------
35 C CALCULATE PRIMITIVE VARIABLES FROM CONSERVATIVE VARIABLES
  uL = momL/rhoL
37  uR = momR/rhoR
  PL = (GAMR-1.0)*(ENL-0.5*rhoL*uL**2.0) - GAMR*PCRITR
39  PR = (GAMR-1.0)*(ENR-0.5*rhoR*uR**2.0) - GAMR*PCRITR
c CALCULATE LEFT AND RIGHT SPEEDS OF SOUND ======================
41  cL = sqrt(GAMR*(PL+PCRITR)/rhoL)
  cR = sqrt(GAMR*(PR+PCRITR)/rhoR)
43 C LITTLE ENERGY AND ENTHALPY
  eL = (PL+GAMR*PCRITR)/((GAMR-1.0)*rhoL)
45  eR = (PR+GAMR*PCRITR)/((GAMR-1.0)*rhoR)
  HL = eL + PL/rhoL + 0.5*uL**2.0
47  HR = eR + PR/rhoR + 0.5*uR**2.0
C CALCULATE ROE AVERAGES ((.)bar ) FOR DENSITY AND PRESSURE =====
49  rhobar = sqrt(rhoL*rhoR)
  call ROEAVG (rhoL,rhoR,uL,uR,ubar)
51  call ROEAVG (rhoL,rhoR,HL,HR,hbar)
  Povrhobar = (PL/sqrt(rhoL) + PR/sqrt(rhoR))/
53    +      (sqrt(rhoL)+sqrt(rhoR)) +
    +      0.5*((uR-uL)/(sqrt(rhoL)+sqrt(rhoR)))**2.0
55  Pbar = rhobar*Povrhobar
C JUST ONE FLUID ------------------------------------------------
57 C cbar = sqrt(GAMR*(Pbar + PCRITR)/rhobar)   !seem to have a
C cbar = sqrt(GAMR*Povrhobar + PCRITR/rhobar) ! choice here..
59  cbar = sqrt((GAMR-1.0)*(hbar-0.5*ubar**2))
C CALCULATE STAR STATES AND WAVE SPEEDS =========================
61  bL = min(uL-cL,ubar-cbar)
  bR = max(ubar+cbar,uR+cR)
63  ustar = (rhoR*uR*(bR-uR) - rhoL*uL*(bL-uL) + PL - PR)/
    +    (rhoR*(bR-uR) - rhoL*(bL-uL))
65  bM = ustar
```

```
      rhoLstar = rhoL*(bL-uL)/(bL-ustar)
      rhoRstar = rhoR*(bR-uR)/(bR-ustar)
      pstar = PL + rhoL*(uL-bL)*(uL-ustar)
      ENLstar = (pstar + GAMR*PCRITR)/(GAMR-1.0) +
     +    0.5*rhoLstar*ustar**2.0
      pstar = PR + rhoR*(bR-uR)*(ustar-uR)
      ENRstar = (pstar + GAMR*PCRITR)/(GAMR-1.0) +
     +    0.5*rhoRstar*ustar**2.0
C ===============================================================
C ---------------------------------------------------------------
C CALCULATE FLUXES DIRECTLY!!
      subsonicflag = .TRUE.
      IF (bL .GT. 0.0) THEN
        fl1 = rhoL*uL
        fl2 = fl1*uL + PL
        fl3 = uL*(PL + ENL)
        subsonicflag = .FALSE.
      END IF
      IF (bR .LT. 0.0) THEN
        fl1 = rhoR*uR
        fl2 = fl1*uR + PR
        fl3 = uR*(PR + ENR)
        subsonicflag = .FALSE.
      END IF
      IF (subsonicflag) THEN
        IF (bM .GT. 0.0) THEN
          fl1 = rhoL*uL + bL*(rhoLstar - rhoL)
          fl2 = rhoL*uL*uL +
     +          bL*(rhoLstar*ustar - rhoL*uL) + PL
          fl3 = uL*(PL+ENL) + bL*(ENLstar - ENL)
        ELSE
          fl1 = rhoR*uR + bR*(rhoRstar - rhoR)
          fl2 = rhoR*uR*uR +
     +          bR*(rhoRstar*ustar - rhoR*uR) + PR
          fl3 = uR*(PR+ENR) + bR*(ENRstar - ENR)
        END IF
      END IF
C ===============================================================
C ===============================================================
      RETURN
      END
C ---------------------------------------------------------------
C ###############################################################
C <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
C ###############################################################
      SUBROUTINE ROEAVG (roL,roR,ftnL,ftnR,RAVG)

      IMPLICIT NONE
C ---------------------------------------------------------------
      DOUBLE PRECISION roL,roR,ftnL,ftnR,RAVG
C ---------------------------------------------------------------
      RAVG = (sqrt(roL)*ftnL + sqrt(roR)*ftnR)/
     +      (sqrt(roL)+sqrt(roR))
C ---------------------------------------------------------------
      RETURN
      END
C ---------------------------------------------------------------
```

SETTSTEP.f

```
      SUBROUTINE SETTSTEP

      INCLUDE "commonblock"
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
```

```
C CHANGE RECORD:
C   20-11-2012: CREATED
C -----------------------------------------------------------------
C SUBROUTINE TO CALCULATE THE TIME STEP BASED ON CFL NUMBER, GRID
C SIZE, MAXIMUM VELOCITY AND MAXIMUM SOUND SPEED
C -----------------------------------------------------------------
C -----------------------------------------------------------------
  DOUBLE PRECISION vsmax,vsig(nrmax)
C -----------------------------------------------------------------
C FIND THE BIGGEST SIGNAL SPEED --------------------------------
  DO i=1,nr
    IF (alpha(i) .LE. 0 ) THEN
      vsig(i) = u(i)+sqrt(gm(1)*(p(i)+Pcr(1))/rho(i))
    ELSE
      vsig(i) = u(i)+sqrt(gm(2)*(p(i)+Pcr(2))/rho(i))
    END IF
  END DO
  vsmax = maxval(vsig,pmask)
C -----------------------------------------------------------------
C SET THE TIME STEP ---------------------------------------------
  dt = CFL*dr/(vsmax)

c REMOVE THIS WHEN FINISHED DEBUGGING w.r.t. 2D code
c dt = 1e-5
c
C -----------------------------------------------------------------
  RETURN
  END
C -----------------------------------------------------------------
```

SETUP.f

```
  SUBROUTINE SETUP

  INCLUDE "commonblock"
C -----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   20-11-2012: CREATED
C   05-08-2013: A MYRIAD OF CHANGES, UNRECORDED
C -----------------------------------------------------------------
C SUBROUTINE TO READ THE INPUT FILES FOR PROGRAM KINGEULER1D
C AND TO SET THE INITIAL FIELDS. THIS SUBROUTINE IS A BIT MESSY.
C -----------------------------------------------------------------
C -----------------------------------------------------------------
  DOUBLE PRECISION rmin
  CHARACTER spare
C READ THE MAIN PARAMETER FILE ----------------------------------
  READ (1,*) spare
  READ (1,*) nr
  READ (1,*) nrout
  READ (1,*) nt
  READ (1,*) outfreq
  READ (1,*) Rd
  READ (1,*) CFL
  READ (1,*) soltype
  READ (1,*) inttype
  READ (1,*) obtype
  READ (1,*) intype
  READ (1,*) coordsno
C -----------------------------------------------------------------
C HOW MANY CELLS FOR BOUNDARIES? 5 IS BE SAFE
C NEEDS TO BE >= 1 + ORDER OF SCHEME
  gfmbuffer = 1 + soltype
C -----------------------------------------------------------------
```

```
C SET THE COEFFICIENTS FOR TIME INTEGRATION ----------------------
  IF (inttype .EQ. 1) THEN
C   1st ORDER EULER
    sfrac(1,1) = 1.0
    sfrac(2,1) = 0.0
    sfrac(3,1) = 1.0
  ELSE IF (inttype .EQ. 2) THEN
C   2nd ORDER TVD (thanks to Shu & Osher)
    sfrac(1,1) = 1.0
    sfrac(2,1) = 0.0
    sfrac(3,1) = 1.0
    sfrac(1,2) = 0.5
    sfrac(2,2) = 0.5
    sfrac(3,2) = 0.5
  ELSE IF (inttype .EQ. 3) THEN
C   3rd ORDER TVD (thanks to Shu & Osher)
    sfrac(1,1) = 1.0
    sfrac(2,1) = 0.0
    sfrac(3,1) = 1.0
    sfrac(1,2) = 0.75
    sfrac(2,2) = 0.25
    sfrac(3,2) = 0.25
    sfrac(1,3) = 1.0/3.0
    sfrac(2,3) = 2.0/3.0
    sfrac(3,3) = 2.0/3.0
  END IF
C SET THE INITIAL FIELDS -----------------------------------------
C WANT TO MODEL SOME KIND OF SHOCK PROBLEM - WHAT TYPE ----------
  OPEN(unit=51,file="SOD.dat",status='old') !Sod shock tube
  OPEN(unit=52,file="AG.dat",status='old') !Air gun bubble
  OPEN(unit=53,file="UE.dat",status='old') !underwater explosion
  OPEN(unit=54,file="SEDOV.dat",status='old') !sedoiv explosion
C ---------------------------------------------------------------
  READ (intype,*) spare
  READ (intype,*) rho0
  READ (intype,*) u0
  READ (intype,*) p0
  READ (intype,*) gamma0
  READ (intype,*) pc0
  READ (intype,*) rho1
  READ (intype,*) u1
  READ (intype,*) p1
  READ (intype,*) gamma1
  READ (intype,*) pc1
  READ (intype,*) Rint
  Pinf = p1
  gm(1) = gamma0
  gm(2) = gamma1
  pcr(1) = pc0
  pcr(2) = pc1
C CREATE THE MESH
  call MESHGEN
C ===============================================================
C OUTPUTTING AND GIVING AN INITIAL VALUE TO INTERFACE INDEX -----
  ofc1 = outfreq + 1
  t=0.0
  DO i=1,nr-1
    IF (r(i) .LT. Rint .AND. r(i+1) .GE. Rint) THEN
      q = i
    END IF
  END DO
C SET UP INITIAL FIELDS ------------------------------------------
  DO i=1,q
    rho(i) = rho0
    u(i) = u0
    P(i) = p0
```

```
    gamm(i) = gamma0
    Pc(i) = pc0
    E(i) = (P(i)+Pc(i)*gamm(i))/(gamm(i)-1)
     +       + 0.5*rho(i)*u(i)**2
    alpha(i) = r(i) - Rint
  END DO
  DO i=q+1,nr+inttype-1
    rho(i) = rho1
    u(i) = u1
    P(i) = p1
    gamm(i) = gamma1
    Pc(i) = pc1
    E(i) = (P(i)+Pc(i)*gamm(i))/(gamm(i)-1)
     +       + 0.5*rho(i)*u(i)**2
    alpha(i) = r(i) - Rint
  END DO
C ---------------------------------------------------------------
C ===============================================================
C CREATE AN ARRAY CALLED pmask FOR SOME HISTORIC (FORGOTTEN)
C REASON... COULD BE FOR SETTING CFL..
  DO i=1,nr-1
    pmask(i) = .TRUE.
  END DO
  DO i=nr+1,nrmax
    pmask(i) = .FALSE.
  END DO
C ===============================================================
  RETURN
  END
C ---------------------------------------------------------------
C 000000000000000000000000000000000000000000000000000000000000000000
  SUBROUTINE MESHGEN

  INCLUDE "commonblock"
C ---------------------------------------------------------------
C CREATE THE MESH BASED ON Rd and nr
C ---------------------------------------------------------------
C SOMETIMES PRESSURE WAVES CONVERGING ON ORIGIN ARE TOO STRONG AT
C r=0, IN WHICH CASE USE rmin TO SHIFT BOUNDARY OUT A BIT (CHEAT)
  rmin = 0.0
  dr = (Rd)/float(nr-1)
  DO i=1,nr+inttype-1
    r(i) = rmin + dr*float(i-1) + 0.5*dr
  END DO
C ---------------------------------------------------------------
  RETURN
  END
C 000000000000000000000000000000000000000000000000000000000000000000
```

SLOPELIMITER.f

```
  SUBROUTINE SLOPELIMITER (sltype,x,B)

  IMPLICIT NONE
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   ??-05-2013: CREATED
C   03-02-2014:   MODIFIED WITH GODUNOV OPTION
C ---------------------------------------------------------------
C A SLOPE LIMITER SUBROUTINE FOR USE WITH MUSCL SCHEME
C ---------------------------------------------------------------
C ---------------------------------------------------------------
  INTEGER sltype
  DOUBLE PRECISION x,B
```

```
C ---------------------------------------------------------------
  IF (sltype .EQ. 0) THEN
    B = 0
  ELSE
C   Monotonised Central
C   B = max(0.0,min(2.0*x,0.5*(1.0+x),2.0))
C   MINMOD
    B = max(0.0,min(1.0,x))
C   Superbee
C   B = max(0.0,min(2.0*x,1.0),min(x,2.0))
  END IF
C ---------------------------------------------------------------
  RETURN
  END
C ---------------------------------------------------------------
```

SPLITFIELDS.f

```
  SUBROUTINE SPLITFIELDS

  INCLUDE "commonblock"
C ---------------------------------------------------------------
C SUBROUTINE TO SPLIT THE FIELDS FOR THE GHOST FLUID METHOD
C ---------------------------------------------------------------
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   20-07-2013: CREATED
C   06-08-2013: VARIOUS CHANGES TO INTERFACE INP STATES
C   07-08-2013: MODIFIED TO WORK ON CONSERVATIVE VARS
C   14-08-2013: ONLY USE NECESSARY GHOST CELLS
C   06-11-2013: OPTION FOR W-INTERP GFM INPUTS
C ---------------------------------------------------------------
C ---------------------------------------------------------------
C ---------------------------------------------------------------
  DOUBLE PRECISION  rhointL,rhointR,uintL,uintR,pintL,pintR
  DOUBLE PRECISION  spr1
C
  DOUBLE PRECISION  drobydrL,dpbydrL,dubydrL
  DOUBLE PRECISION  drobydrR,dpbydrR,dubydrR
C
  DOUBLE PRECISION  RTL,RTR
C
C ---------------------------------------------------------------
C ===============================================================
C CHOOSE THE INPUT STATES TO THE INTERFACE RIEMANN PROBLEM AND
C THE LIMITS FOR THE STATES WHICH ARE MODIFIED.
C   intL and intR are the cells either side of the
C     interface;
C   intLin and intRin are the cells for the input states
C     of the Riemann problem;
C   intLabs and intRabs are the indices of the last real
C     cells not modified with the GFM.
C
  rLOW(1) = 1
  rHIGH(1) = q+1+gfmbuffer
  rLOW(2) = q-gfmbuffer
  rHIGH(2) = nr+inttype-1
C
C IN THE MAIN PROGRAM, PRESSURE IS ADJUSTED TO BE NON-NEGATIVE
C EVERY TIME-STEP. ENERGY IS NOT ADJUSTED. IF SPLITFIELDS
C TAKES P FOR INPUTS RESULTS WILL DIFFER FROM IF SPLITFIELDS
C TAKES E FOR INPUTS.
C
  rhointL = rho(q)
```

```
      uintL = u(q)
      rhointR = rho(q+1)
      uintR = u(q+1)
      pintL = (gamma0-1.0)*(E(q)-0.5*rhointL*uintL*uintL)-
     +      gamma0*pc0
      pintR = (gamma1-1.0)*(E(q+1)-0.5*rhointR*uintR*uintR)-
     +      gamma1*pc1
C -------------------------------------------------------------
      IF (.FALSE.) THEN
C WEIGHTED INTERPOLATION!!! ------------------------------------
      RTL = Rint - 0.5*dr !or maybe Rint - 0.5*dr
      RTR = Rint + 0.5*dr !or maybe Rint + 0.5*dr
      call GFMWINTERP (rho(q-3),rho(q-2),rho(q-1),rho(q),
     +    r(q-3),r(q-2),r(q-1),r(q),Rint,RTL,dr,rhointL)
      call GFMWINTERP (rho(q+4),rho(q+3),rho(q+2),rho(q+1),
     +    r(q+4),r(q+3),r(q+2),r(q+1),Rint,RTR,-1.0*dr,rhointR)
      call GFMWINTERP (u(q-3),u(q-2),u(q-1),u(q),
     +    r(q-3),r(q-2),r(q-1),r(q),Rint,RTL,dr,uintL)
      call GFMWINTERP (u(q+4),u(q+3),u(q+2),u(q+1),
     +    r(q+4),r(q+3),r(q+2),r(q+1),Rint,RTR,-1.0*dr,uintR)
      call GFMWINTERP (p(q-3),p(q-2),p(q-1),p(q),
     +    r(q-3),r(q-2),r(q-1),r(q),Rint,RTL,dr,pintL)
      call GFMWINTERP (p(q+4),p(q+3),p(q+2),p(q+1),
     +    r(q+4),r(q+3),r(q+2),r(q+1),Rint,RTR,-1.0*dr,pintR)
      END IF
C -------------------------------------------------------------
C SOLVE THE RIEMANN PROBLEM AT THE INTERFACE
      call RIEMANNINTERFACE (rhointL,pintL,uintL,gamma0,Pc0,
     +        rhointR,pintR,uintR,gamma1,Pc1,
     +        rhoLI,rhoRI,UI,PI,SL,SR)
C =============================================================
c LAST GHOST CELL APPLIES A REFLECTIVE BOUNDARY IN THE GHOST
C BAND. THIS ALLOWS CHEAP CALCULATIONS OF CONSERVATION PROPERTIES
C WITHIN GHOST BANDS, AND HENCE IN EACH FLUID.
C FLUID 0 -----------------------------------------------------
C FLUID 0: REAL CELLS -----------------------------------------
      DO i=1,q
        m0(1,1,i) = rho(i)
        m0(1,2,i) = rho(i)*u(i)
        m0(1,3,i) = E(i)
        m0(1,4,i) = p(i)
        m0(1,5,i) = u(i)
      END DO
C FLUID 0: ORDINARY GHOST CELLS -------------------------------
      DO i=q+1,q+gfmbuffer
        m0(1,1,i) = rhoLI
        m0(1,4,i) = pI
        m0(1,5,i) = uI
        m0(1,2,i) = m0(1,1,i)*m0(1,5,i)
        m0(1,3,i) = (m0(1,4,i)+gamma0*Pc0)/
     +      (gamma0-1.0)+0.5*(m0(1,2,i)**2)/m0(1,1,i)
      END DO
C FLUID 0: LAST GHOST CELL - APPLICATION OF REFLECTIVE BC -------
      m0(1,1,q+1+gfmbuffer) = m0(1,1,q+gfmbuffer)
      m0(1,2,q+1+gfmbuffer) = -1.0*m0(1,2,q+gfmbuffer)
      m0(1,3,q+1+gfmbuffer) = m0(1,3,q+gfmbuffer)
      m0(1,4,q+1+gfmbuffer) = m0(1,4,q+gfmbuffer)
      m0(1,5,q+1+gfmbuffer) = -1.0*m0(1,5,q+gfmbuffer)
C FLUID 1 -----------------------------------------------------
C FLUID 1: REAL CELLS -----------------------------------------
      DO i=q+1,nr+inttype-1
        m0(2,1,i) = rho(i)
        m0(2,2,i) = rho(i)*u(i)
        m0(2,3,i) = E(i)
        m0(2,4,i) = p(i)
        m0(2,5,i) = u(i)
```

```
      END DO
C FLUID 1: ORDINARY GHOST CELLS --------------------------------
      DO i=q+1-gfmbuffer,q
        m0(2,1,i) = rhoRI
        m0(2,4,i) = pI
        m0(2,5,i) = uI
        m0(2,2,i) = m0(2,1,i)*m0(2,5,i)
        m0(2,3,i) = (m0(2,4,i)+gamma1*Pc1)/
     +      (gamma1-1.0)+0.5*(m0(2,2,i)**2)/m0(2,1,i)
      END DO
C FLUID 1: LAST GHOST CELL - APPLICATION OF REFLECTIVE BC -------
      m0(2,1,q-gfmbuffer) = m0(2,1,q+1-gfmbuffer)
      m0(2,2,q-gfmbuffer) = -1.0*m0(2,2,q+1-gfmbuffer)
      m0(2,3,q-gfmbuffer) = m0(2,3,q+1-gfmbuffer)
      m0(2,4,q-gfmbuffer) = m0(2,4,q+1-gfmbuffer)
      m0(2,5,q-gfmbuffer) = -1.0*m0(2,5,q+1-gfmbuffer)
C ================================================================
      RETURN
      END
C ================================================================
C 0000000000000000000000000000000000000000000000000000000000000000
C ================================================================
      SUBROUTINE RIEMANNINTERFACE (rhoL,pL,uL,GAML,PCRITL,rhoR,pR,uR,
     +           GAMR,PCRITR,
     +           rhoLstar,rhoRstar,ustar,pstar,
     +           bL,bR)
      IMPLICIT NONE
C ----------------------------------------------------------------
C SUBROUTINE TO SOLVE A RIEMANN PROBLEM - ROE TYPE
C ----------------------------------------------------------------
C SIMPLY RETURNS THE STAR STATES
c http://dx.doi.org/10.1016/j.jcp.2009.06.002 (sort of)
C ----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   15-07-2013: CREATED
C   25-07-2013: EMBEDDED IN SPLITFIELDS.f
C ----------------------------------------------------------------
      DOUBLE PRECISION  rhoL,PL,uL,GAML,PCRITL,rhoR,PR,uR,
     +         GAMR,PCRITR
C
      DOUBLE PRECISION  eL,eR,HL,HR,GL,GR,PHIL,PHIR
      DOUBLE PRECISION  pstar,ustar,rhoLstar,rhoRstar
      DOUBLE PRECISION  ENLstar,ENRstar
      DOUBLE PRECISION  cL,cR
      DOUBLE PRECISION  bL,bR,bM
      DOUBLE PRECISION  ubar,cbar,rhobar,Povrhobar,Pbar,hbar
      DOUBLE PRECISION  PHIbar,Gbar
      DOUBLE PRECISION  Gdebar,PHIdrhobar
C ----------------------------------------------------------------
c CALCULATE LEFT AND RIGHT SPEEDS OF SOUND ======================
      cL = sqrt(GAML*(PL+PCRITL)/rhoL)
      cR = sqrt(GAMR*(PR+PCRITR)/rhoR)
C CALCULATE LEFT AND RIGHT ENERGY, ENTHALPY, etc ================
      eL = (PL+GAML*PCRITL)/((GAML-1.0)*rhoL)
      eR = (PR+GAMR*PCRITR)/((GAMR-1.0)*rhoR)
      HL = eL + PL/rhoL + 0.5*uL**2.0
      HR = eR + PR/rhoR + 0.5*uR**2.0
      GL = GAML-1.0
      GR = GAMR-1.0
      PHIL = (GAML-1.0)*eL
      PHIR = (GAMR-1.0)*eR
C CALCULATE ROE AVERAGES ((.)bar ) FOR PROPERTIES ===============
      rhobar = sqrt(rhoL*rhoR)
      call ROEAVG (rhoL,rhoR,uL,uR,ubar)
```

```
      call ROEAVG (rhoL,rhoR,HL,HR,hbar)
      Povrhobar = (PL/sqrt(rhoL) + PR/sqrt(rhoR))/
     +      (sqrt(rhoL)+sqrt(rhoR)) +
     +      0.5*((uR-uL)/(sqrt(rhoL)+sqrt(rhoR)))**2
      Pbar = rhobar*Povrhobar
C WE ARE LOOKING AT 2 FLUIDS! I KNOW THIS BECAUSE WE ALWAYS ARE -
        call ROEAVG (rhoL,rhoR,GL,GR,Gbar)
        call ROEAVG (rhoL,rhoR,PHIL,PHIR,PHIbar)
        cbar = sqrt(PHIbar + Gbar*Povrhobar)
C -------------------------------------------------------------
C CALCULATE STAR STATES AND WAVE SPEEDS ========================
      bL = min(uL-cL,ubar-cbar)
      bR = max(ubar+cbar,uR+cR)
      ustar = (rhoR*uR*(bR-uR) + rhoL*uL*(uL-bL) + PL - PR)/
     +    (rhoR*(bR-uR) + rhoL*(uL-bL))
      bM = ustar
      pstar = PL + rhoL*(uL-bL)*(uL-ustar)
C   pstar = PR + rhoR*(bR-uR)*(ustar-uR)
      rhoLstar = rhoL*(bL-uL)/(bL-ustar)
      rhoRstar = rhoR*(bR-uR)/(bR-ustar)
      ENLstar = (pstar + GAML*PCRITL)/(GAML-1.0) +
     +    0.5*rhoLstar*ustar**2
      ENRstar = (pstar + GAMR*PCRITR)/(GAMR-1.0) +
     +    0.5*rhoRstar*ustar**2
C =============================================================
C =============================================================
C APPROXIMATE SOLVER - LINEAR
C cbar = 0.5*(cL+cR)
C rhobar = 0.5*(rhoL+rhoR)
C pstar = 0.5*(PL+PR) - 0.5*(UR-UL)*rhobar*cbar
C ustar = 0.5*(uL+uR) - 0.5*(PR-PL)/(rhobar*cbar)
C rhoLstar = rhoL + (uL-ustar)*rhobar/cbar
C rhoRstar = rhoR + (ustar-uR)*rhobar/cbar
C =============================================================
C -------------------------------------------------------------
      RETURN
      END
C -------------------------------------------------------------
C -------------------------------------------------------------
C #############################################################
C =============================================================
      SUBROUTINE GFMWINTERP (prop1,prop2,prop3,prop4,
     +    r1,r2,r3,r4,Ri,Rt,delr,phiT)

      IMPLICIT NONE
C -------------------------------------------------------------
C -------------------------------------------------------------
C SUBROUTINE TO DO WEIGHTED INTERPOLATION FOR GFM INPUT STATES
C THE PURPOSE OF THIS SUBROUTINE IS TO CREATE GFM INPUT STATES
C WHICH VARY SMOOTHLY EVEN AS THE INTERFACE (AND HENCE INPUTS TO
C   THIS SUBROUTINE) PASSES A CELL CENTRE. IT DOESN'T IMPROVE THE
C RESULTS...
C -------------------------------------------------------------
C -------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   06-11-2013: CREATED
C -------------------------------------------------------------
C INPUTS
      DOUBLE PRECISION prop1,prop2,prop3,prop4
      DOUBLE PRECISION r1,r2,r3,r4,Ri,Rt,delr
C INTERNAL
      DOUBLE PRECISION phiE(3)
      DOUBLE PRECISION x,w1,w2,w3
      INTEGER i
```

```
C OUTPUTS
      DOUBLE PRECISION phiT
C -------------------------------------------------------------
C CALCULATE THE THREE ESTIMATES
      phiE(1) = prop1 + (prop2-prop1)*(rt-r1)/delr
      phiE(2) = prop2 + (prop3-prop2)*(rt-r2)/delr
      phiE(3) = prop3 + (prop4-prop3)*(rt-r3)/delr
C CALCULATE THE WEIGHTS!
      x = (Ri-r4)/delr
      w1 = x - x**2.0
      w3 = x - x**2.0
      w2 = 1.0-(w1+w3)
C CALCULATE THE WEIGHTED ESTIMATE
      phiT = w1*phiE(1) + w2*phiE(2) + w3*phiE(3)
      RETURN
      END
C -------------------------------------------------------------
C =============================================================
```

WENOZ.f

```
      SUBROUTINE WENOZ (fim2,fim1,fi,fip1,fip2,fimh,fiph)

      IMPLICIT NONE
C -------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   ??-03-2013: CREATED
C -------------------------------------------------------------
C SUBROUTINE TO CALCULATE FLUXES USING BORGES 5TH ORDER MODIFIED
C WENO SCHEME (as mentioned in Hu et al 2009 - requires a trail
C of references to be followed.)
C -------------------------------------------------------------
C -------------------------------------------------------------
C INPUT FUNCTION
      DOUBLE PRECISION  fim2,fim1,fi,fip1,fip2
C OUTPUT WENOZ APPROXIMATION
      DOUBLE PRECISION  fimh,fiph
C SMOOTHNESS INDICATORS
      DOUBLE PRECISION  beta0,beta1,beta2
      DOUBLE PRECISION  tau5,eps
      DOUBLE PRECISION  beta0z,beta1z,beta2z
C WEIGHTINGS
      DOUBLE PRECISION  d0,d1,d2
      DOUBLE PRECISION  alpha0z,alpha1z,alpha2z,sumalphaz
      DOUBLE PRECISION  w0z,w1z,w2z
C SMALL STENCIL INTERPOLATIONS
      DOUBLE PRECISION  fimh0,fimh1,fimh2,fiph0,fiph1,fiph2
C -------------------------------------------------------------
C SET BASIC WEIGHTINGS ----------------------------------------
      d0=0.3
      d1=0.6
      d2=0.1
C -------------------------------------------------------------
C CALCULATE INITIAL SMOOTHNESS INDICATORS ----------------------
      beta0 = (13.0/12.0)*(fim2-2.0*fim1+fi)**2.0 +
     +    0.25*(fim2-4.0*fim1+3.0*fi)**2.0
      beta1 = (13.0/12.0)*(fim1-2.0*fi+fip1)**2.0 +
     +    0.25*(fim1-fip1)**2.0
      beta2 = (13.0/12.0)*(fi-2.0*fip1+fip2)**2.0 +
     +    0.25*(3.0*fi-4.0*fip1+fip2)**2.0
      tau5 = abs(beta0-beta2)
C -------------------------------------------------------------
C CALCULATE BETTER SMOOTHNESS INDICATORS ----------------------
```

```
      eps = 1e-40
      beta0z = (beta0+eps)/(beta0+tau5+eps)
      beta1z = (beta1+eps)/(beta1+tau5+eps)
      beta2z = (beta2+eps)/(beta2+tau5+eps)
C -----------------------------------------------------------------
C CALCULATE DIFFERENT WEIGHTINGS -------------------------------
      alpha0z = d0/beta0z
      alpha1z = d1/beta1z
      alpha2z = d2/beta2z
      sumalphaz = alpha0z + alpha1z + alpha2z
C -----------------------------------------------------------------
C CALCULATE FINAL WEIGHTINGS -----------------------------------
      w0z = alpha0z/sumalphaz
      w1z = alpha1z/sumalphaz
      w2z = alpha2z/sumalphaz
C -----------------------------------------------------------------
C CALCULATE THREE 3 POINT STENCIL ESTIMATES AT i+0.5 ------------
      fiph0 = fim2/3.0 - 7.0*fim1/6.0 + 11.0*fi/6.0
      fiph1 = -1.0*fim1/6.0 + 5.0*fi/6.0 + fip1/3.0
      fiph2 = fi/3.0 + 5.0*fip1/6.0 - fip2/6.0
C -----------------------------------------------------------------
C AND A 5th ORDER ESTIMATE USING THE FINAL WEIGHTS --------------
      fiph = w0z*fiph0 + w1z*fiph1 + w2z*fiph2
C -----------------------------------------------------------------
C CALCULATE THREE 3 POINT STENCIL ESTIMATES AT i-0.5 ------------
      fimh0 = -1.0*fim2/6.0 + 5.0*fim1/6.0 + fi/3.0
      fimh1 = fim1/3.0 + 5.0*fi/6.0 - fip1/6.0
      fimh2 = 11.0*fi/6.0 - 7.0*fip1/6.0 + fip2/3.0
C -----------------------------------------------------------------
C AND A 5th ORDER ESTIMATE USING THE FINAL WEIGHTS --------------
      fimh = w0z*fimh0 + w1z*fimh1 + w2z*fimh2
C -----------------------------------------------------------------
      RETURN
      END
```

init.params

```
# MAIN PARAMETER FILE. DIRECTS KE1D TO SPECIFIC INITIAL CONDS' #
200             :nr
200             :nrout
50000             :nt
200             :outfreq
1.0             :Rd
0.8             :CFL
1             :soltype(1-Godunov,2-Muscl,5-WENO-Z)
1             :inttype(1-euler,2-2nd order,3-3rd order tvd)
3             :obtype(1-nr,2-dudt=0,3-nlaa)
52              :intype(51=SOD,52=AG,53=UE,54=SEDOV)
2.0             :coordsno(0.0-Ca,1.0-cy,2.0-po)
```

AG.dat

```
# INITIAL DATA FOR SEISMIC AIR GUN PROBLEM #
102.0         :rhoL
0.0         :uL
8.85e6          :pL
1.4         :gammaL
0.0         :PcL
1000.0          :rhoR
0.0         :uR
1.77e5          :pR
7         :gammaR
3e8         :PcR
0.1         :Rint
```

SOD.dat

```
# INITIAL DATA FOR AIR-HELIUM SOD SHOCK PROBLEM #
1.0        :rhoL
0.0        :uL
1.0        :pL
1.4        :gammaL
0.0        :PcL
0.125        :rhoR
0.0        :uR
0.1        :pR
1.66667        :gammaR
0.0        :PcR
0.5        :Rint
```

UE.dat

```
# INITIAL DATA FOR UNDERWATER EXPLOSION PROBLEM #
1.63         :rhoL
0.0        :uL
83810.0        :pL
1.4        :gammaL
0.0        :PcL
1.025        :rhoR
0.0        :uR
10.0         :pR
5.5        :gammaR
4921.15        :PcR
0.16         :Rint
```

E.2 Two-dimensional code

This section contains all the source code and parameter files required to generate the two-dimensional results shown in the thesis. The source code consists of 16 FORTRAN source files and a file containing a list of common variables. The source code is listed alphabetically according to filename, followed by two parameter files.

commonblock

```
C -------------------------------------------------------------
  PARAMETER (nx1max=802,nx2max=202,ntmax=100002,npmax=10)
C
C GRID
  INTEGER nx1,nx2,bcount,bcell(nx1max,nx2max),
     +    bci(1000),bcj(1000),q(nx2max)
  DOUBLE PRECISION  x1d,x2d,x1(nx1max),x2(nx2max),dx1,
     +        dx2,coordsno,x(nx1max,nx2max),
     +        z(nx1max,nx2max),r(nx1max,nx2max),
     +        th(nx1max,nx2max)
  DOUBLE PRECISION  V(nx1max,nx2max),Sx1mh(nx1max,nx2max),
     +        Sx2mh(nx1max,nx2max)
  COMMON  / grid /
     +    x1d,x2d,x1,x2,dx1,dx2,coordsno,x,z,r,th,V,Sx1mh,Sx2mh,
     +    nx1,nx2,bcount,bcell,bci,bcj,q
C FLAGS_ETC
```

```
      INTEGER outfreq,ofc1,viscflag,ghostflag,boundflag,distflag
      COMMON  / flags_etc /
     +    outfreq,ofc1,viscflag,ghostflag,boundflag,distflag
C TIME
      INTEGER tn,nt
      DOUBLE PRECISION  dt,t,CFL,lc(nx1max,nx2max)
      COMMON  / time /
     +    tn,nt,dt,t,CFL,lc
C INITIAL FIELD
      DOUBLE PRECISION  ux11,ux12,ux21,ux22,rho1,rho2,p1,p2,
     +      gm(2),pc(2),Rint
      COMMON  / initial_properties /
     +    ux11,ux12,ux21,ux22,rho1,rho2,p1,p2,gm,pc,Rint
C BOUNDARY
      DOUBLE PRECISION  Pinf(nx2max),Pinf0(nx2max),grav,
     +      f(ntmax),df(ntmax),d2f(ntmax),
     +      thist(ntmax),ubold,dgun
      COMMON  / boundary /
     +    Pinf,Pinf0,grav,f,df,d2f,thist,ubold,dgun
C PROPERTIES
      DOUBLE PRECISION  rho(nx1max,nx2max),ux1(nx1max,nx2max),
     +    ux2(nx1max,nx2max),P(nx1max,nx2max),E(nx1max,nx2max),
     +    vort(nx1max,nx2max),Uprop(2,npmax,nx1max,nx2max),
     +    dUprop(npmax,nx1max,nx2max),dUmu(npmax,nx1max,nx2max),
     +    IMPmu(npmax,nx1max,nx2max)
      COMMON  / flow_properties /
     +    rho, ux1, ux2, P,E,vort,Uprop,dUprop,dUmu,IMPmu
C LEVEL SET
      DOUBLE PRECISION  alpha(nx1max,nx2max),
     +      normx1(nx1max,nx2max),normx2(nx1max,nx2max),
     +      ulsx1(nx1max,nx2max),ulsx2(nx1max,nx2max)
      COMMON  / level_set /
     +    alpha,normx1,normx2,ulsx1,ulsx2
C
      PARAMETER (pie=3.14159)
C ---------------------------------------------------------------
```

DUMPRESULTS.f

```
      SUBROUTINE DUMPRESULTS

      INCLUDE "commonblock"
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   20-11-2012: CREATED
C   ??<08-2013: MODIFIED FOR FAR FIELD OUTPUT IF REQ
C   ??<04-2014: MODIFIED FOR THE 2D CODE...
C ---------------------------------------------------------------
C SUBROUTINE TO WRITE THE OUTPUTS IF IT IS THE REQUIRED TIME
C ---------------------------------------------------------------
      INTEGER ofspc
      CHARACTER dmpfile*(14),tron*(5)
      DOUBLE PRECISION sp1,sp2,sp3,sp4
      DOUBLE PRECISION v_cell,m_cell,summz,summ,b_mass,b_z,rplus
C EITHER WRITE TO OUTPUT OR ADD 1 TO THE OUTPUT COUNTER ---------
      IF (ofc1 .GE. outfreq) THEN
C   IT IS TIME TO WRITE THE OUTPUT
      WRITE (6,*) "WRITING"
c   what is the vorticity?
      call VORTICITY
C   WRITE (6,*) "-------- t = ",t
      write(tron,'(i5.5)') tn
      tron=adjustl(tron)
      dmpfile="DUMP"//tron//".DUMP"
```

```
28      OPEN(2,file=dmpfile)
        write(2,*) t
30      DO i=1,nx1
          DO j=1,nx2
32          WRITE(7,*) rho(i,j)
            WRITE(8,*) p(i,j)
34          WRITE(9,*) ux1(i,j)
            WRITE(10,*) ux2(i,j)
36          WRITE(11,*) E(i,j)
            WRITE(12,*) alpha(i,j)
38          WRITE(13,*) ux1(i,j)*sin(x2(j))+
         +          ux2(i,j)*cos(x2(j))
40          WRITE(14,*) ux1(i,j)*cos(x2(j))-
         +          ux2(i,j)*sin(x2(j))
42          WRITE(15,*) vort(i,j)
            WRITE(16,*) IMPmu(1,i,j),IMPmu(2,i,j),
44       +          IMPmu(3,i,j),IMPmu(4,i,j)
            WRITE(2,*) rho(i,j),p(i,j),ux1(i,j),
46       +          ux2(i,j),E(i,j),alpha(i,j)
          END DO
48      END DO
        close(2)
50  C   RESET THE OUTPUT COUNTER
        ofc1 = 1
52    ELSE
    C   ADD 1 TO THE OUTPUT COUNTER
54      ofc1 = ofc1 + 1
      END IF
56  C Writing other stuff
      WRITE (20,*) t,dt !time step
58    WRITE (21,*) t,Rint ! interface location
      sp1 = 0.0
60    sp2 = 0.0
      sp3 = 0.0
62    sp4 = 0.0
      DO j=2,nx2-1
64      sp1 = sp1+p(nx1,j)
        sp2 = sp2+p(q(j),j)*sin(x2(j))*x1(q(j))**2.0
66      sp3 = sp3 + sin(x2(j))*x1(q(j))**2.0
        sp4 = sp4+ux1(q(j),j)*sin(x2(j))*x1(q(j))**2.0
68    END DO
      sp1 = sp1/float(nx2-2)
70  ! sp2 = sp2/float(nx2-2)
      sp2 = sp2/sp3
72    WRITE(22,*) t,p(nx1,25) ! boundary pressure
      WRITE(23,*) t,sp2 ! interface pressure
74    WRITE(26,*) t,rho(nx1,25)
      WRITE(27,*) t,ux1(nx1,25)
76    summz=0.0
      summ=0.0
78    DO j=1,nx2
        DO i=1,nx1-1
80        IF (alpha(i,j).LT.0)THEN
          IF (alpha(i,j)*alpha(i+1,j).LE.0.0)THEN
82          q(j)=i
          END IF
84        Rint = x1(q(j)) - alpha(q(j),j)/
         +         max(1.0,-1.0*alpha(q(j),j)/dx1)
86        IF (alpha(q(j),j).Eq.0.0)THEN
            Rint = x1(q(j))
88        ELSE IF (alpha(q(j)+1,j).EQ.0.0)THEN
            Rint = x1(q(j)+1)
90        END IF
          IF (alpha(i,j)*alpha(i+1,j).LE.0.0)THEN
92          rplus = Rint
          ELSE
```

```
        rplus = x1(i)+0.5*dx1
      END IF
        v_cell = (1.0/3.0)*((rplus)**
     +        3.0 - (x1(i)-0.5*dx1)**3.0)*
     +        (cos(x2(j)-0.5*dx2)-cos(x2(j)+0.5*dx2))
     +        *2.0*pie
        m_cell = rho(i,j)*v_cell
!       m_cell = v_cell
        summz = summz + m_cell*z(i,j)
        summ = summ + m_cell
      END IF
    END DO
  END DO
  b_z = summz/summ
  b_mass = summ
  WRITE(24,*) t,b_mass
  WRITE(25,*) t,b_z
  RETURN
  END
```

EVOLVE.f

```
  SUBROUTINE EVOLVE

  INCLUDE "commonblock"
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   20-07-2013: CREATED
C   07-08-2013: SPHERICALITY BY SOURCE TERMS
C   07-08-2013: MODIFIED TO WORK ON CONSERVATIVE VARS
C   13-08-2013: MUSCL SCHEME IS NOW 2-STEP(ie. CORRECT)
C   14-08-2013: STOPPED SOLVING DUPLICATE RIEMANN PROBS
C   14-08-2013: ONLY USE NECESSARY GHOST CELLS..
C   13-09-2013: MUSCL RECONSTRUCTION OF PRIMITIVE VARS
C   25-09-2013: WENO-Z RECONSTRUCTION OPTION!
C   25-09-2013: 3rd ORDER TVD TIME INTEGRATION OPTION!
C ---------------------------------------------------------------
C THE 1 PHASE EULER SOLVER. CHOOSES A FIELD TO WORK ON. RECONSTR-
C UCTS THE FIELD. CALCULATES FLUXES. UPDATES THE VALUES. ADDS THE
C SOURCE TERMS. UPDATES THE VALUES OF THE CHOSEN FIELD.
C ---------------------------------------------------------------
  INTEGER x1LOW,x1HIGH,x2LOW,x2HIGH
  DOUBLE PRECISION cosij,sinij,sro,sux1,sux2
c viscous bits!
  INTEGER im1,ip1,jm1,jp1
  DOUBLE PRECISION divu,mu,taux1x1(nx1max,nx2max),
     +    taux1x2(nx1max,nx2max),
     +    taux2x2(nx1max,nx2max),vis(4,nx1max,nx2max)
  DOUBLE PRECISION uij,uip1,uim1,ujp1,ujm1,vij,vip1,vim1,vjp1,
     +      vjm1
  DOUBLE PRECISION dudr,dudth,dvdr,dvdth,costh,sinth
  DOUBLE PRECISION strp,strrho,strT,sT0,smu0,sC
  x1LOW = 1
  x1HIGH = nx1
  x2LOW = 1
  x2HIGH = nx2
C FOR EACH FLUID!!
  DO k=1,2
C   HOW SHALL WE SWEEP?
C   APPLY BC IN X1 DIRECTION
    DO j=1,nx2
      Uprop(k,1,1,j) = Uprop(k,1,2,j)
      Uprop(k,2,1,j) = -1.0*Uprop(k,2,2,j)
      Uprop(k,3,1,j) = Uprop(k,3,2,j)
```

```
      Uprop(k,4,1,j) = Uprop(k,4,2,j)
46    END DO
      IF (boundflag .EQ. 1) THEN
48      IF (ghostflag .EQ. 0) THEN
          call NLAABC(k)
50      ELSE
          call NLAABCGHOST(k)
52      END IF
      ELSE
54      DO j=1,nx2
          Uprop(k,1,nx1,j)=Uprop(k,1,nx1-1,j)
56        Uprop(k,2,nx1,j)=-1.*Uprop(k,2,nx1-1,j)
          Uprop(k,3,nx1,j)=Uprop(k,3,nx1-1,j)
58        Uprop(k,4,nx1,j)=Uprop(k,4,nx1-1,j)
        END DO
60    END IF
C     EVOLVE IN X1 DIRECTION
62    call SWEEP1D (k,1)
C     APPLY BC IN X2 DIRECTION
64    DO i=1,nx1
        Uprop(k,1,i,1) = Uprop(k,1,i,2)
66      Uprop(k,2,i,1) = Uprop(k,2,i,2)
        Uprop(k,3,i,1) = -1.0*Uprop(k,3,i,2)
68      Uprop(k,4,i,1) = Uprop(k,4,i,2)
C
70      Uprop(k,1,i,nx2) = Uprop(k,1,i,nx2-1)
        Uprop(k,2,i,nx2) = Uprop(k,2,i,nx2-1)
72      Uprop(k,3,i,nx2) = -1.0*Uprop(k,3,i,nx2-1)
        Uprop(k,4,i,nx2) = Uprop(k,4,i,nx2-1)
74    END DO
C     EVOLVE IN X2 DIRECTION
76    call SWEEP1D (k,2)
C     GRAVITY SOURCE TERMS!!
78    IF (grav .NE. 0.0) THEN
      DO i=2,nx1-1
80      DO j=2,nx2-1
          cosij = cos(x2(j))
82        sinij = sin(x2(j))
          sro = Uprop(k,1,i,j)
84        sux1 = Uprop(k,2,i,j)/sro
          sux2 = Uprop(k,3,i,j)/sro
86        Uprop(k,2,i,j)=Uprop(k,2,i,j)-dt*sro*
       +          grav*cosij
88        Uprop(k,3,i,j)=Uprop(k,3,i,j)+dt*sro*
       +          grav*sinij
90        Uprop(k,4,i,j)=Uprop(k,4,i,j)-dt*sro*
       +        grav*(cosij*sux1 - sinij*sux2)
92      END DO
      END DO
94    END IF
    END DO
96  IF (viscflag .EQ. 1) THEN
c   VISCOUS TERMS!!!
98  DO k=1,2
      IF (k.EQ.1)THEN
100     mu = 2e-5
        sT0 = 273.15
102     sC = 110.4
        smu0 = 1.716e-5
104   ELSE
        mu = 1.0D-3
106   END IF
      DO i=1,nx1
108   DO j=1,nx2
        im1 = max(1,i-1)
110     ip1 = min(nx1,i+1)
```

```
      jm1 = max(1,j-1)
      jp1 = min(nx2,j+1)
c     set costh and sinth
      costh = cos(x2(j))
      sinth = sin(x2(j))
c     set the u,v for here and next door
      uij = Uprop(k,2,i,j)/Uprop(k,1,i,j)
      uip1 = Uprop(k,2,ip1,j)/Uprop(k,1,ip1,j)
      uim1 = Uprop(k,2,im1,j)/Uprop(k,1,im1,j)
      ujp1 = Uprop(k,2,i,jp1)/Uprop(k,1,i,jp1)
      ujm1 = Uprop(k,2,i,jm1)/Uprop(k,1,i,jm1)
      vij = Uprop(k,3,i,j)/Uprop(k,1,i,j)
      vip1 = Uprop(k,3,ip1,j)/Uprop(k,1,ip1,j)
      vim1 = Uprop(k,3,im1,j)/Uprop(k,1,im1,j)
      vjp1 = Uprop(k,3,i,jp1)/Uprop(k,1,i,jp1)
      vjm1 = Uprop(k,3,i,jm1)/Uprop(k,1,i,jm1)
c     calculate the dudr,dudth,dvdr,dvdth
      IF (i.EQ.1)THEN
        dudr = (uip1-uij)/dx1
        dvdr = (vip1-vij)/dx1
      ELSE IF (i.EQ.nx1) THEN
        dudr = (uij-uim1)/dx1
        dvdr = (vij-vim1)/dx1
      ELSE
        dudr = (uip1-uim1)/(2.0*dx1)
        dvdr = (vip1-vim1)/(2.0*dx1)
      END IF
      IF (j.EQ.1)THEN
        dudth = (ujp1-uij)/dx2
        dvdth = (vjp1-vij)/dx2
      ELSE IF (j.EQ.nx2) THEN
        dudth = (uij-ujm1)/dx2
        dvdth = (vih-vjm1)/dx2
      ELSE
        dudth = (ujp1-ujm1)/(2.0*dx2)
        dvdth = (vjp1-vjm1)/(2.0*dx2)
      END IF
c     calculate divergence of velocity field..
      divu = dudr + 2.0*uij/x1(i) + dvdth/x1(i) +
     +        vij*costh/(x1(i)*sinth)
c     calculate stress tensor components...
      IF (k.EQ.1.AND.alpha(i,j).LE.0)THEN
        strrho = Uprop(k,1,i,j)
        strp = (gm(1)-1.0)*(Uprop(k,4,i,j) -
     +          0.5*strrho*(uij**2 +vij**2))
        strT = strp/(287.0*strrho)
        mu = smu0*((sT0+sC)/(strT+sC))*
     +        (strT/sT0)**(3.0/2.0)
      END IF
      taux1x1(i,j)=mu*(2.0*dudr - 2.0*divu/3.0)
      taux1x2(i,j)=mu*(dvdr-(vij/x1(i))+dudth/x1(i))
      taux2x2(i,j)=mu*(2.0*(dvdth/x1(i)+uij/x1(i)) -
     +        2.0*divu/3.0)
    END DO
    END DO
    DO i=2,nx1-1
    DO j=2,nx2-1
c     set costh and sinth
      costh = cos(x2(j))
      sinth = sin(x2(j))
c     calculate viscous terms....
      vis(1,i,j) = 0.0
      vis(2,i,j) = (taux1x1(i+1,j)-taux1x1(i-1,j))
     +        /(2.0*dx1) + 2.0*taux1x1(i,j)/x1(i)
     +        + (taux1x2(i,j+1)-taux1x2(i,j-1))/
     +        (2.0*dx2*x1(i)) +
```

```
     +        taux1x2(i,j)*costh/(x1(i)*sinth)
      vis(3,i,j) = (taux1x2(i+1,j)-taux1x2(i-1,j))/
     +        (2.0*dx1) + 2.0*taux1x2(i,j)/x1(i)
     +        + (taux2x2(i,j+1)-taux2x2(i,j-1))/
     +        (2.0*dx2*x1(i)) +
     +        taux2x2(i,j)*costh/(x1(i)*sinth)
      vis(4,i,j) = (uip1*taux1x1(i+1,j)+
     +        vip1*taux1x2(i+1,j) -
     +        uim1*taux1x1(i-1,j)-
     +        vim1*taux1x2(i-1,j))/(2.0*dx1)
     +        + 2.0*(uij*taux1x1(i,j)+
     +        vij*taux1x2(i,j))/x1(i) +
     +        (ujp1*taux1x2(i,j+1)+
     +        vjp1*taux2x2(i,j+1)-
     +        ujm1*taux1x2(i,j-1)-
     +        vjm1*taux2x2(i,j-1))/(2.0*dx2*x1(i))
     +        + (uij*taux1x2(i,j)+vij*taux2x2(i,j))
     +        *costh/(x1(i)*sinth)
    END DO
    END DO
    DO l=1,4
    DO i=2,nx1-1
    DO j=2,nx2-1
      Uprop(k,l,i,j) = Uprop(k,l,i,j) + dt*vis(l,i,j)
      dUmu(l,i,j)=dt*vis(l,i,j)
    END DO
    END DO
    END DO
  END DO
  END IF
  RETURN
  END
```

FINDPAIR.f

```
  SUBROUTINE FINDPAIR (iA,jA,iB,jB)

  INCLUDE "commonblock"
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   04-03-2014: CREATED
C ---------------------------------------------------------------
C SPLIT THE FIELDS USING A GFM
C ---------------------------------------------------------------
C DECLARATIONS!
  INTEGER ia,ja,ib,jb
  INTEGER im1,ip1,jm1,jp1
  INTEGER tc(2,8)
  DOUBLE PRECISION spim,spip,spjm,spjp
  DOUBLE PRECISION spimjm,spimjp,spipjm,spipjp
  DOUBLE PRECISION nax,naz,nbx,nbz
  DOUBLE PRECISION storeA,minangle
  im1 = max(1,ia-1)
  ip1 = min(nx1,ia+1)
  jm1 = max(1,ja-1)
  jp1 = min(nx2,ja+1)
  tc(1,1) = ip1
  tc(2,1) = ja
  tc(1,2) = ip1
  tc(2,2) = jm1
  tc(1,3) = ia
  tc(2,3) = jm1
  tc(1,4) = im1
  tc(2,4) = jm1
```

```
32    tc(1,5) = im1
      tc(2,5) = ja
34    tc(1,6) = im1
      tc(2,6) = jp1
36    tc(1,7) = ia
      tc(2,7) = jp1
38    tc(1,8) = ip1
      tc(2,8) = jp1
40    minangle = 1e12
  C Loop over all neighbours
42    DO k=1,8
  C     Check whether the potential neighbour cell is a bound-
44  c   -ary cell of the right type..
      IF (bcell(tc(1,k),tc(2,k)).NE.0.AND.
46     +    bcell(tc(1,k),tc(2,k)).NE.bcell(ia,ja))THEN
  C     It's a potential pair. Is it the best choice?
48      nax = normx1(ia,ja)*sin(x2(ja)) +
       +      normx2(ia,ja)*cos(x2(ja))
50      naz = normx1(ia,ja)*cos(x2(ja)) -
       +      normx2(ia,ja)*sin(x2(ja))
52      nbx = normx1(tc(1,k),tc(2,k))*sin(x2(tc(2,k))) +
       +      normx2(tc(1,k),tc(2,k))*cos(x2(tc(2,k)))
54      nbz = normx1(tc(1,k),tc(2,k))*cos(x2(tc(2,k))) -
       +      normx2(tc(1,k),tc(2,k))*sin(x2(tc(2,k)))
56      storeA = nax*nbx + naz*nbz
  c     storeA = normx1(ia,ja)*normx1(tc(1,k),tc(2,k)) +
58  c      +      normx2(ia,ja)*normx2(tc(1,k),tc(2,k))
        IF (abs(1.0-storeA).LE.abs(1.0-minangle))THEN
60  C     If it's the closest yet then it'll do for now!
          minangle = storeA
62        ib = tc(1,k)
          jb = tc(2,k)
64      END IF
      END IF
66    END DO
  C write(6,*) ia,ja,ib,jb,bcell(ia,ja)
68    if (ib.lt.1.or.ib.gt.nx1.or.jb.lt.1.or.jb.gt.nx2)then
        write(6,*) "FIND PAIR BAD MATCH!!!",ia,ja,ib,jb
70      call FINISH(2)
      end if
72    RETURN
      END
```

FINISH.f

```
1     SUBROUTINE FINISH(error_number)

3     IMPLICIT NONE
  C ---------------------------------------------------------------
5 C AUTHOR: J. R. C. KING
  C ---------------------
7 C CHANGE RECORD:
  C   19-06-2014: CREATED
9 C ---------------------------------------------------------------
  C SUBROUTINE TO STOP THE PROGRAM...
11c ---------------------------------------------------------------
      INTEGER error_number,i
13    WRITE (6,*) "KE2D TERMINATING"
      IF (error_number .EQ. 0) THEN
15      WRITE (6,*) "NORMALLY"
      END IF
17    IF (error_number .EQ. 1) THEN
        WRITE (6,*) "ERROR CODE = ",error_number
19      WRITE (6,*) "BAD INPUT FILE REQUESTED"
      END IF
21    IF (error_number .EQ. 2) THEN
```

```
      WRITE (6,*) "ERROR CODE = ",error_number
      WRITE (6,*) "FAILURE TO CORRECTLY FIND PAIRS"
      WRITE (6,*) "CLOSING TO PREVENT SEGMENTATION FAULT"
    END IF
    WRITE (6,*) "CLOSING FILES"
    CLOSE(1)
    DO i=7,16
      CLOSE(i)
    END DO
    DO i=20,26
    CLOSE(i)
    END DO
    DO i=30,33
      CLOSE(i)
    END DO
    CLOSE (41)
    WRITE (6,*) "FINISHED. GOOD BYE!"
    STOP
    END
```

GFMSPLIT.f

```
    SUBROUTINE GFMSPLIT

    INCLUDE "commonblock"
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   04-03-2014: CREATED
C ---------------------------------------------------------------
C SPLIT THE FIELDS USING A GFM - QUITE AN ORDEAL IN 2D...
C ---------------------------------------------------------------
C DECLARATIONS!
    INTEGER ib,jb
    INTEGER im1,ip1,jm1,jp1
    DOUBLE PRECISION spim,spip,spjm,spjp
    DOUBLE PRECISION spimjm,spimjp,spipjm,spipjp
    DOUBLE PRECISION storeA,minangle
    DOUBLE PRECISION rhoA,rhoB,uA,uB,pA,pB,uAt,uBt,ugt
    DOUBLE PRECISION rhoLI,rhoRI,PI,UI,sp1,sp2
    DOUBLE PRECISION ddx1(9,nx1max,nx2max),ddx2(9,nx1max,nx2max)
    DOUBLE PRECISION advcfl

    advcfl = 1.0*min(dx1,dx2)
C Put copies of states into both domains for now. We will soon
C overwrite everything in ghost regions...
    DO i=1,nx1
      DO j=1,nx2
        Uprop(1,1,i,j)=rho(i,j)
        Uprop(1,2,i,j)=rho(i,j)*ux1(i,j)
        Uprop(1,3,i,j)=rho(i,j)*ux2(i,j)
        Uprop(1,4,i,j)=E(i,j)
        Uprop(1,5,i,j)=p(i,j)
        Uprop(1,6,i,j)=ux1(i,j)
        Uprop(1,7,i,j)=ux2(i,j)
        Uprop(2,1,i,j)=rho(i,j)
        Uprop(2,2,i,j)=rho(i,j)*ux1(i,j)
        Uprop(2,3,i,j)=rho(i,j)*ux2(i,j)
        Uprop(2,4,i,j)=E(i,j)
        Uprop(2,5,i,j)=p(i,j)
        Uprop(2,6,i,j)=ux1(i,j)
        Uprop(2,7,i,j)=ux2(i,j)
      END DO
    END DO
    DO k=1,bcount
```

```
      i=bci(k)
      j=bcj(k)
C for each type of boundary cell...
      call FINDPAIR(i,j,ib,jb)
      rhoA = rho(i,j)
      pA = p(i,j)
      uA = ux1(i,j)*normx1(i,j) +
     +      ux2(i,j)*normx2(i,j)
!     uAt = sqrt(ux1(i,j)**2.0 +
!     +      ux2(i,j)**2.0 - uA**2.0)
      uAt = ux1(i,j)*normx2(i,j)-
     +      ux2(i,j)*normx1(i,j)
      rhoB = rho(ib,jb)
      pB = p(ib,jb)
      uB = ux1(ib,jb)*normx1(ib,jb) +
     +      ux2(ib,jb)*normx2(ib,jb)
!     uBt = sqrt(ux1(ib,jb)**2.0 +
!     +      ux2(ib,jb)**2.0 - uB**2.0)
      uBt = ux1(ib,jb)*normx2(ib,jb)-
     +      ux2(ib,jb)*normx1(ib,jb)
c       !!   for viscosity...
      IF(viscflag.EQ.0)THEN
        ugt = ubt
      ELSE
        ugt = uat
      END IF
c !!!!
      IF (bcell(i,j) .EQ. 2) THEN
        call RIEMANNINTERFACE (
     +      rhoB,pB,uB,gm(1),pc(1),
     +      rhoA,pA,uA,gm(2),pc(2),
     +      rhoLI,rhoRI,UI,PI,sp1,sp2
     +      )
c       write(6,*) rhoLI,rhoRI,uI,pI
c ! comment the next 2 lines for rGFM
c  ! leave in for mGFM(ish)
c ! copying pressures from air, rGFM for all else
        PI = pB
c
        Uprop(1,1,i,j) = rhoLI
        Uprop(1,5,i,j) = PI
        Uprop(1,8,i,j) = UI
        Uprop(1,9,i,j) = ugt
        Uprop(1,6,i,j) = UI*normx1(i,j)
     +        +ugt*normx2(i,j)
        Uprop(1,7,i,j) = UI*normx2(i,j)
     +        -ugt*normx1(i,j)
        Uprop(1,2,i,j)=rhoLI*Uprop(1,6,i,j)
        Uprop(1,3,i,j)=rhoLI*Uprop(1,7,i,j)
        Uprop(1,4,i,j)=(PI+gm(1)*pc(1))/
     +      (gm(1)-1.0) + 0.5*rhoLI*UI**2.0
      ELSE IF (bcell(i,j) .EQ. 1) THEN
        call RIEMANNINTERFACE (
     +      rhoA,pA,uA,gm(1),pc(1),
     +      rhoB,pB,uB,gm(2),pc(2),
     +      rhoLI,rhoRI,UI,PI,sp1,sp2
     +      )
c ! comment the next line for rGFM
c  ! leave in for mGFM(ish)
c ! Copying pressures from air. rGFM for all else...
        PI = pA
c
        Uprop(2,1,i,j) = rhoRI
        Uprop(2,5,i,j) = PI
        Uprop(2,8,i,j) = UI
        Uprop(2,9,i,j) = ugt
```

```
      Uprop(2,6,i,j) = UI*normx1(i,j)
     +        +ugt*normx2(i,j)
      Uprop(2,7,i,j) = UI*normx2(i,j)
     +        -ugt*normx1(i,j)
      Uprop(2,2,i,j)=rhoRI*Uprop(2,6,i,j)
      Uprop(2,3,i,j)=rhoRI*Uprop(2,7,i,j)
      Uprop(2,4,i,j)=(PI+gm(2)*pc(2))/
     +      (gm(2)-1.0) + 0.5*rhoRI*UI**2.0
    END IF
  END DO
C ADVECT PROPERTIES - use upwind difference scheme...
  DO k=1,int(nx1/5)
    DO i=1,nx1
      im1 = max(i-1,1)
      ip1 = min(i+1,nx1)
      DO j=1,nx2
        jm1 = max(j-1,1)
        jp1 = min(j+1,nx2)
        IF (alpha(i,j).GT.0.AND.
     +        bcell(i,j).EQ.0)THEN
C       If in fluid 2 region and beyond bcells
        DO l=1,9
          IF (normx1(i,j).GT.0.0) THEN
C         Left sided x1 deriv
            ddx1(l,i,j) =
     +            (Uprop(1,l,i,j)-
     +            Uprop(1,l,im1,j))/dx1
          ELSE
C         right sided x1 deriv
            ddx1(l,i,j) =
     +            (Uprop(1,l,ip1,j)-
     +            Uprop(1,l,i,j))/dx1
          END IF
          IF (normx2(i,j).GT.0.0) THEN
C         Left sided x2 deriv
            ddx2(l,i,j) =
     +            (Uprop(1,l,i,j)-
     +            Uprop(1,l,i,jm1))/dx2
          ELSE
C         right sided x2 deriv
            ddx2(l,i,j) =
     +            (Uprop(1,l,i,jp1)-
     +            Uprop(1,l,i,j))/dx2
          END IF
        END DO
        END IF
      END DO
    END DO
    DO i=1,nx1
    DO j=1,nx2
      IF (alpha(i,j).GT.0.AND.
     +      bcell(i,j).EQ.0)THEN
        DO l=1,9
          Uprop(1,l,i,j)=Uprop(1,l,i,j)-
     +          advcfl*(normx1(i,j)*
     +          ddx1(l,i,j) + normx2(i,j)*
     +          ddx2(l,i,j)/x1(i))
        END DO
c
        Uprop(1,2,i,j) = Uprop(1,1,i,j)*
     +          Uprop(1,6,i,j)
        Uprop(1,3,i,j) = Uprop(1,1,i,j)*
     +          Uprop(1,7,i,j)
        Uprop(1,4,i,j) = (Uprop(1,5,i,j)+
     +          gm(1)*pc(1))/
     +        (gm(1)-1.0) + 0.5*Uprop(1,1,i,j)*
```

```
     +        (Uprop(1,6,i,j)**2.0 +
     +        Uprop(1,7,i,j)**2.0)
      END IF
    END DO
    END DO
    DO i=1,nx1
      im1 = max(i-1,1)
      ip1 = min(i+1,nx1)
      DO j=1,nx2
        jm1 = max(j-1,1)
        jp1 = min(j+1,nx2)
        IF (alpha(i,j).LE.0.AND.
     +        bcell(i,j).EQ.0)THEN
C       If in fluid 1 region and beyond bcells
        DO l=1,9
          IF (normx1(i,j).LE.0.0) THEN
C         Left sided x1 deriv
            ddx1(l,i,j) =
     +             (Uprop(2,l,i,j)-
     +             Uprop(2,l,im1,j))/dx1
          ELSE
C         right sided x1 deriv
            ddx1(l,i,j) =
     +             (Uprop(2,l,ip1,j)-
     +             Uprop(2,l,i,j))/dx1
          END IF
          IF (normx2(i,j).LE.0.0) THEN
C         Left sided x2 deriv
            ddx2(l,i,j) =
     +             (Uprop(2,l,i,j)-
     +             Uprop(2,l,i,jm1))/dx2
          ELSE
C         right sided x2 deriv
            ddx2(l,i,j) =
     +             (Uprop(2,l,i,jp1)-
     +             Uprop(2,l,i,j))/dx2
          END IF
        END DO
        END IF
      END DO
    END DO
    DO i=1,nx1
    DO j=1,nx2
      IF (alpha(i,j).LE.0.AND.
     +      bcell(i,j).EQ.0)THEN
        DO l=1,9
          Uprop(2,l,i,j)=Uprop(2,l,i,j)+
     +          advcfl*(normx1(i,j)*
     +          ddx1(l,i,j) + normx2(i,j)*
     +          ddx2(l,i,j)/x1(i))
        END DO
c
        Uprop(2,2,i,j) = Uprop(2,1,i,j)*
     +        Uprop(2,6,i,j)
        Uprop(2,3,i,j) = Uprop(2,1,i,j)*
     +        Uprop(2,7,i,j)
        Uprop(2,4,i,j) = (Uprop(2,5,i,j)+
     +        gm(2)*pc(2))/
     +        (gm(2)-1.0) + 0.5*Uprop(2,1,i,j)*
     +        (Uprop(2,6,i,j)**2.0 +
     +        Uprop(2,7,i,j)**2.0)
      END IF
    END DO
    END DO
  END DO
  DO i=1,nx1
```

```
      DO j=1,nx2
244     IF (alpha(i,j).LE.0)THEN
          ulsx1(i,j) = Uprop(2,6,i,j)
246       ulsx2(i,j) = Uprop(2,7,i,j)
        ELSE
248       ulsx1(i,j) = Uprop(1,6,i,j)
          ulsx2(i,j) = Uprop(1,7,i,j)
250     END IF
c       ulsx1(i,j) = ux1(i,j)
252 c       ulsx2(i,j) = ux2(i,j)
      END DO
254   END DO
    RETURN
256   END
    C ---------------------------------------------------------------
258   SUBROUTINE RIEMANNINTERFACE (rhoL,pL,uL,GAML,PCRITL,rhoR,pR,uR,
     +          GAMR,PCRITR,
260  +          rhoLstar,rhoRstar,ustar,pstar,
     +          bL,bR)
262   IMPLICIT NONE
    C ---------------------------------------------------------------
264 C SUBROUTINE TO SOLVE A RIEMANN PROBLEM - HLLC SOLVER
    C ---------------------------------------------------------------
266 C SIMPLY RETURNS THE STAR STATES
    C ---------------------------------------------------------------
268 C AUTHOR: J. R. C. KING
    C ---------------------
270 C CHANGE RECORD:
    C   15-07-2013: CREATED
272 C   25-07-2013: EMBEDDED IN SPLITFIELDS.f
    DOUBLE PRECISION  rhoL,PL,uL,GAML,PCRITL,rhoR,PR,uR,
274  +        GAMR,PCRITR
    C
276   DOUBLE PRECISION  eL,eR,HL,HR,GL,GR,PHIL,PHIR
      DOUBLE PRECISION  pstar,ustar,rhoLstar,rhoRstar
278   DOUBLE PRECISION  ENLstar,ENRstar
      DOUBLE PRECISION  cL,cR
280   DOUBLE PRECISION  bL,bR,bM
      DOUBLE PRECISION  ubar,cbar,rhobar,Povrhobar,Pbar,hbar
282   DOUBLE PRECISION  PHIbar,Gbar
      DOUBLE PRECISION  Gdebar,PHIdrhobar
284 c CALCULATE LEFT AND RIGHT SPEEDS OF SOUND ======================
      cL = sqrt(GAML*(PL+PCRITL)/rhoL)
286   cR = sqrt(GAMR*(PR+PCRITR)/rhoR)
    C CALCULATE LEFT AND RIGHT ENERGY, ENTHALPY, etc ================
288   eL = (PL+GAML*PCRITL)/((GAML-1.0)*rhoL)
      eR = (PR+GAMR*PCRITR)/((GAMR-1.0)*rhoR)
290   HL = eL + PL/rhoL + 0.5*uL**2.0
      HR = eR + PR/rhoR + 0.5*uR**2.0
292   GL = GAML-1.0
      GR = GAMR-1.0
294   PHIL = (GAML-1.0)*eL
      PHIR = (GAMR-1.0)*eR
296 C CALCULATE ROE AVERAGES ((.)bar ) FOR PROPERTIES ===============
      rhobar = sqrt(rhoL*rhoR)
298   call ROEAVG (rhoL,rhoR,uL,uR,ubar)
      call ROEAVG (rhoL,rhoR,HL,HR,hbar)
300   Povrhobar = (PL/sqrt(rhoL) + PR/sqrt(rhoR))/
     +      (sqrt(rhoL)+sqrt(rhoR)) +
302  +      0.5*((uR-uL)/(sqrt(rhoL)+sqrt(rhoR)))**2
      Pbar = rhobar*Povrhobar
304 C WE ARE LOOKING AT 2 FLUIDS! I KNOW THIS BECAUSE WE ALWAYS ARE -
      call ROEAVG (rhoL,rhoR,GL,GR,Gbar)
306   call ROEAVG (rhoL,rhoR,PHIL,PHIR,PHIbar)
      cbar = sqrt(PHIbar + Gbar*Povrhobar)
308 C CALCULATE STAR STATES AND WAVE SPEEDS ==========================
```

```
      bL = min(uL-cL,ubar-cbar)
      bR = max(ubar+cbar,uR+cR)
      ustar = (rhoR*uR*(bR-uR) + rhoL*uL*(uL-bL) + PL - PR)/
     +    (rhoR*(bR-uR) + rhoL*(uL-bL))
      bM = ustar
      pstar = PL + rhoL*(uL-bL)*(uL-ustar)
C     pstar = PR + rhoR*(bR-uR)*(ustar-uR)
      rhoLstar = rhoL*(bL-uL)/(bL-ustar)
      rhoRstar = rhoR*(bR-uR)/(bR-ustar)
      ENLstar = (pstar + GAML*PCRITL)/(GAML-1.0) +
     +    0.5*rhoLstar*ustar**2
      ENRstar = (pstar + GAMR*PCRITR)/(GAMR-1.0) +
     +    0.5*rhoRstar*ustar**2
C APPROXIMATE SOLVER - LINEAR
c cbar = 0.5*(cL+cR)
c rhobar = 0.5*(rhoL+rhoR)
c pstar = 0.5*(PL+PR) - 0.5*(UR-UL)*rhobar*cbar
c ustar = 0.5*(uL+uR) - 0.5*(PR-PL)/(rhobar*cbar)
c rhoLstar = rhoL + (uL-ustar)*rhobar/cbar
c rhoRstar = rhoR + (ustar-uR)*rhobar/cbar
      RETURN
      END
```

KINGEULER2D.f

```
      PROGRAM KINGEULER2D

      INCLUDE "commonblock"
C -----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
      INTEGER qr
      DOUBLE PRECISION store1
      DOUBLE PRECISION dV
C OPEN FILES ----------------------------------------------------
C INPUT FILES = 1
C FIELD OUTPUT FILES = 7,8,9,10,11,12
C TIME VARYING PROPERTY FILES = 20,21
c GRID = 30,31,32,33
C LOG FILE = 5
      OPEN(unit=1,file="init.params",status='old')
      OPEN(unit=7,file="rho.out")
      OPEN(unit=8,file="P.out")
      OPEN(unit=9,file="ux1.out")
      OPEN(unit=10,file="ux2.out")
      OPEN(unit=11,file="E.out")
      OPEN(unit=12,file="A.out")
      OPEN(unit=13,file="ux.out")
      OPEN(unit=14,file="uz.out")
      OPEN(unit=15,file="vort.out")
      OPEN(unit=16,file="IMPmu.out")
      OPEN(unit=20,file="dt.out")
      OPEN(unit=21,file="interface.out")
      OPEN(unit=22,file="pbound.out")
      OPEN(unit=23,file="pint.out")
      OPEN(unit=24,file="bmass.out")
      OPEN(unit=25,file="bz.out")
      OPEN(unit=26,file="rhobound.out")
      OPEN(unit=27,file="ubound.out")
      OPEN(unit=30,file="x1.out")
      OPEN(unit=31,file="x2.out")
      OPEN(unit=32,file="x.out")
      OPEN(unit=33,file="z.out")
C SET SOME INITIAL VARIABLES ------------------------------------
      call SETUP
      call SETTSTEP
```

```
      call LEVELSET(0)
      tn=1
C TIME LOOP ----------------------------------------------------
      DO tn = 1,nt
C     PROGRESS COUNTER
        WRITE (6,*) 'Progress: ',
     +        100.0*float(tn)/float(nt), '%'
C     SET THE TIME STEP
        call SETTSTEP
      ! IF (t.GT. 0.001)THEN !bodge for seeing ghost wave
      ! dt = 1.48D-6
      ! END IF
C     WRITE RESULTS ----------------------------------------
        call DUMPRESULTS
C     SPLIT FIELDS USING THE GFM
        call GFMSPLIT
C     UPDATE THE LEVEL SET!!!
        call LEVELSET(1)
C     FIND FLUXES AND UPDATE IN TIME
        call EVOLVE
C     RECOMBINE FIELDS
        DO i=1,nx1
          DO j=1,nx2
            IF (alpha(i,j) .LT. 0.0) THEN
C     TO DETERMINE MAGNITUDE OF CHANGE...
          dUprop(1,i,j)=Uprop(1,1,i,j)-rho(i,j)
          dUprop(2,i,j)=Uprop(1,2,i,j)-rho(i,j)*ux1(i,j)
          dUprop(3,i,j)=Uprop(1,3,i,j)-rho(i,j)*ux2(i,j)
          dUprop(4,i,j)=Uprop(1,4,i,j)-E(i,j)
            rho(i,j) = Uprop(1,1,i,j)
            ux1(i,j)=Uprop(1,2,i,j)/Uprop(1,1,i,j)
            ux2(i,j)=Uprop(1,3,i,j)/Uprop(1,1,i,j)
            E(i,j) = Uprop(1,4,i,j)
            P(i,j) =  (gm(1)-1)*(E(i,j)-0.5*
     +        rho(i,j)*(ux1(i,j)**2 + ux2(i,j)**2))
     +        -gm(1)*pc(1)
            ELSE
C     TO DETERMINE MAGNITUDE OF CHANGE...
          dUprop(1,i,j)=Uprop(2,1,i,j)-rho(i,j)
          dUprop(2,i,j)=Uprop(2,2,i,j)-rho(i,j)*ux1(i,j)
          dUprop(3,i,j)=Uprop(2,3,i,j)-rho(i,j)*ux2(i,j)
          dUprop(4,i,j)=Uprop(2,4,i,j)-E(i,j)
            rho(i,j) = Uprop(2,1,i,j)
            ux1(i,j)=Uprop(2,2,i,j)/Uprop(2,1,i,j)
            ux2(i,j)=Uprop(2,3,i,j)/Uprop(2,1,i,j)
            E(i,j) = Uprop(2,4,i,j)
            P(i,j) =  (gm(2)-1)*(E(i,j)-0.5*
     +        rho(i,j)*(ux1(i,j)**2 + ux2(i,j)**2))
     +        -gm(2)*pc(2)
            END IF
C     TO DETERMINE MAGNITUDE OF CHANGE
            DO l=1,4
              IMPmu(l,i,j)=abs(dUmu(l,i,j))/
     +          abs(dUprop(l,i,j))
            END DO
          END DO
        END DO
c     END DO
C     UPDATE TIME ------------------------------------------
        t = t + dt
      END DO
C DONE, call FINISH
      call FINISH(0)
      STOP
      END
```

LEVELSET.f

```
      SUBROUTINE LEVELSET(lsv_flag)

      INCLUDE "commonblock"
C -----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   25-02-2014: CREATED
C -----------------------------------------------------------------
C UPDATE A LEVEL SET!!! ----------------------------------------
      INTEGER lsv_flag
      INTEGER im3,im2,im1,ip1,ip2,ip3,jm3,jm2,jm1,jp1,jp2,jp3
      DOUBLE PRECISION dadx1(nx1max,nx2max),dadx2(nx1max,nx2max)
      DOUBLE PRECISION dadx1m,dadx1p,dadx2m,dadx2p
      DOUBLE PRECISION Sp(nx1max,nx2max),dls(nx1max,nx2max)
      DOUBLE PRECISION GLS(nx1max,nx2max),ga,gb,gc,gd,dtau
      DOUBLE PRECISION s,storex1,storex2,mgrada
      DOUBLE PRECISION store1,spare1,spR
C PART 1 - ADVECTION OF LEVEL SET
      DO j=1,nx2
        DO i=4,nx1-3
          IF (ulsx1(i,j) .GT. 0.0) THEN
C           LEFT SIDED DERIVATIVES ----------------
            call WENOZ((alpha(i-2,j)-alpha(i-3,j)),
     +           (alpha(i-1,j)-alpha(i-2,j)),
     +           (alpha(i,j)-alpha(i-1,j)),
     +           (alpha(i+1,j)-alpha(i,j)),
     +           (alpha(i+2,j)-alpha(i+1,j)),
     +           spare1,store1)
            dadx1(i,j) = store1/dx1
!           dadx1(i,j)=(alpha(i,j)-alpha(i-1,j))/dx1 !1st order!
          ELSE
C           RIGHT SIDED DERIVATIVES ---------------
            call WENOZ((alpha(i+3,j)-alpha(i+2,j)),
     +           (alpha(i+2,j)-alpha(i+1,j)),
     +           (alpha(i+1,j)-alpha(i,j)),
     +           (alpha(i,j)-alpha(i-1,j)),
     +           (alpha(i-1,j)-alpha(i-2,j)),
     +           spare1,store1)
            dadx1(i,j) = store1/dx1
!           dadx1(i,j)=(alpha(i+1,j)-alpha(i,j))/dx1 !1st order!
          END IF
        END DO
        DO i=2,3
          IF (ulsx1(i,j) .GT. 0.0) THEN
C           LEFT SIDED DERIVATIVES ----------------
            dadx1(i,j) = (alpha(i,j)-alpha(i-1,j))
     +           /dx1
          ELSE
C           RIGHT SIDED DERIVATIVES ---------------
            dadx1(i,j) = (alpha(i+1,j)-alpha(i,j))
     +           /dx1
          END IF
        END DO
        DO i=nx1-2,nx1-1
          IF (ulsx1(i,j) .GT. 0.0) THEN
C           LEFT SIDED DERIVATIVES ----------------
            dadx1(i,j) = (alpha(i,j)-alpha(i-1,j))
     +           /dx1
          ELSE
C           RIGHT SIDED DERIVATIVES ---------------
            dadx1(i,j) = (alpha(i+1,j)-alpha(i,j))
     +           /dx1
          END IF
        END DO
```

```
      dadx1(1,j) = dadx1(2,j)
      dadx1(nx1,j) = dadx1(nx1-1,j)
    END DO
    DO i=1,nx1
      DO j=4,nx2-3
        IF (ulsx2(i,j) .GT. 0.0) THEN
C         LEFT SIDED DERIVATIVES ----------------
          call WENOZ((alpha(i,j-2)-alpha(i,j-3)),
     +           (alpha(i,j-1)-alpha(i,j-2)),
     +           (alpha(i,j)-alpha(i,j-1)),
     +           (alpha(i,j+1)-alpha(i,j)),
     +           (alpha(i,j+2)-alpha(i,j+1)),
     +           spare1,store1)
          dadx2(i,j) = store1/dx2
! dadx2(i,j) = (alpha(i,j)-alpha(i,j-1))/dx2 !1st order!
        ELSE
C         RIGHT SIDED DERIVATIVES ---------------
          call WENOZ((alpha(i,j+3)-alpha(i,j+2)),
     +           (alpha(i,j+2)-alpha(i,j+1)),
     +           (alpha(i,j+1)-alpha(i,j)),
     +           (alpha(i,j)-alpha(i,j-1)),
     +           (alpha(i,j-1)-alpha(i,j-2)),
     +           spare1,store1)
          dadx2(i,j) = store1/dx2
! dadx2(i,j) = (alpha(i,j-1)-alpha(i,j))/dx2 !1st order!
        END IF
      END DO
      DO j=2,3
        IF (ulsx2(i,j) .GT. 0.0) THEN
C         LEFT SIDED DERIVATIVES ----------------
          dadx2(i,j) = (alpha(i,j)-alpha(i,j-1))
     +           /dx2
        ELSE
C         RIGHT SIDED DERIVATIVES ---------------
          dadx2(i,j) = (alpha(i,j+1)-alpha(i,j))
     +           /dx2
        END IF
      END DO
      DO j=nx2-2,nx2-1
        IF (ulsx2(i,j) .GT. 0.0) THEN
C         LEFT SIDED DERIVATIVES ----------------
          dadx2(i,j) = (alpha(i,j)-alpha(i,j-1))
     +           /dx2
        ELSE
C         RIGHT SIDED DERIVATIVES ---------------
          dadx2(i,j) = (alpha(i,j+1)-alpha(i,j))
     +           /dx2
        END IF
      END DO
C     CONSTRAINT: theta-derivatives of level set equal zero
c     at theta-boundaries. Physically, this means the bubble
C     doesn't have any corners - which is as it should be!
      dadx2(i,1) = 0.0
      dadx2(i,nx2) = 0.0
    END DO
    IF (lsv_flag.EQ.1)THEN
    DO i=1,nx1
      DO j=1,nx2
        alpha(i,j) = alpha(i,j) - dt*
     +       (ulsx1(i,j)*dadx1(i,j)+ulsx2(i,j)*dadx2(i,j)
     +       /x1(i))
      END DO
    END DO
    j=1
    DO i=1,nx1
      alpha(i,j) = alpha(i,j+1)
```

```
132   END DO
      i=1
134   DO j=1,nx2
        alpha(i,j) = alpha(i+1,j)
136   END DO
      j=nx2
138   DO i=1,nx1
        alpha(i,j) = alpha(i,j-1)
140   END DO
      END IF
142 C PART 2: RE-INITIALISATION OF THE LEVEL SET!
    C Based on the scheme by Russo & Smereka (2000), JCP 163, p51-67
144 C Or maybe on Spelt (2005), JCP 207, p389-404.
    C Label all boundary cells!
146   call SETBCELL
      DO i=1,nx1
148     DO j=1,nx2
    C     1. Calculate the sign of the level set at
150 c     each node. "Mollified" sign function.
        sp(i,j) = alpha(i,j)/sqrt(alpha(i,j)**2.0 +
152    +        dx1**2.0)
    c     2. Set dLS(i,j) to the value of the level set.
154     dLS(i,j) = alpha(i,j)
      END DO
156   END DO
      dtau = 0.1*x1(1)*dx1*dx2/(x1(1)*dx2+dx1)
158   DO k=1,int(2*nx1/5)
    C   3. Calculate GLS, upwind disc' of abs(grad(alpha))-1
160     DO i=1,nx1
        im1 = max(1,i-1)
162     ip1 = min(nx1,i+1)
        DO j=1,nx2
164       jm1 = max(1,j-1)
          jp1 = min(nx2,j+1)
166       ga = (alpha(i,j)-alpha(im1,j))/dx1
          gb = (alpha(ip1,j)-alpha(i,j))/dx1
168       gc = (alpha(i,j)-alpha(i,jm1))/(dx2*x1(i))
          gd = (alpha(i,jp1)-alpha(i,j))/(dx2*x1(i))
170       IF (alpha(i,j).GT.0)THEN
            GLS(i,j) = sqrt(max((max(ga,0.0)**2),
172    +        (min(gb,0.0)**2))+max((max(gc,0.0)**2),
       +        (min(gd,0.0)**2)))-1.0
174       ELSE
            GLS(i,j) = sqrt(max((min(ga,0.0)**2),
176    +        (max(gb,0.0)**2))+max((min(gc,0.0)**2),
       +        (max(gd,0.0)**2)))-1.0
178       END IF
        END DO
180     END DO
    C   4. Update the values of dLS
182     DO i=1,nx1
        DO j=1,nx2
184       IF (bcell(i,j) .EQ. 0)THEN
    C     Not a bcell, normal reinitialisation!
186         dLS(i,j) = dLS(i,j) - dtau*sp(i,j)*
       +          GLS(i,j)
188       ELSE
    C     A bcell, special reinitialisation??
190 c   or just leave it be!!!
          END IF
192     END DO
        END DO
194   END DO
      IF (lsv_flag.EQ.1)THEN
196   DO i=1,nx1
      DO j=1,nx2
```

```
198       alpha(i,j) = dLS(i,j)
        END DO
200     END DO
        j=1
202     DO i=1,nx1
          alpha(i,j) = alpha(i,j+1)
204     END DO
        i=1
206     DO j=1,nx2
          alpha(i,j) = alpha(i+1,j)
208     END DO
        j=nx2
210     DO i=1,nx1
          alpha(i,j) = alpha(i,j-1)
212     END DO
        END IF
214   C PART 3: CALCULATION OF NORMALS
        call SETBCELL
216     DO i=2,nx1-1
          DO j=2,nx2-1
218         dadx1(i,j) = (alpha(i+1,j)-alpha(i-1,j))/
           +         (2.0*dx1)
220         dadx2(i,j) = (alpha(i,j+1)-alpha(i,j-1))/
           +         (2.0*dx2)
222         spare1 = sqrt(dadx1(i,j)**2.0 +
           +         (dadx2(i,j)/x1(i))**2.0)
224         normx1(i,j) = dadx1(i,j)/spare1
            normx2(i,j) = dadx2(i,j)/spare1
226       END DO
        END DO
228     DO i=2,nx1-1
          normx1(i,1) = normx1(i,2)
230       normx1(i,nx2) = normx1(i,nx2-1)
          normx2(i,1) = normx2(i,2)
232       normx2(i,nx2) = normx2(i,nx2-1)

234     END DO
        DO j=2,nx2-1
236       normx1(1,j) = normx1(2,j)
          normx1(nx1,j) = normx1(nx1-1,j)
238       normx2(1,j) = normx2(2,j)
          normx2(nx1,j) = normx2(nx1-1,j)
240     END DO
        normx1(1,1) = normx1(2,2)
242     normx2(1,1) = normx2(2,2)
        normx1(nx1,1) = normx1(nx1-1,2)
244     normx2(nx1,1) = normx2(nx1-1,2)
        normx1(1,nx2) = normx1(2,nx2-1)
246     normx2(1,nx2) = normx2(2,nx2-1)
        normx1(nx1,nx2) = normx1(nx1-1,nx2-1)
248     normx2(nx1,nx2) = normx2(nx1-1,nx2-1)
      C Calculate the bubble volume and hence an effective radius
250     spR = 0.0
        DO j=1,nx2
252       DO i=1,nx1-1
            IF (alpha(i,j)*alpha(i+1,j).LE.0.0)THEN
254           q(j)=i
            END IF
256       END DO
          Rint = x1(q(j)) - alpha(q(j),j)/
258        +      max(1.0,-1.0*alpha(q(j),j)/dx1)
          IF (alpha(q(j),j).Eq.0.0)THEN
260         Rint = x1(q(j))
          ELSE IF (alpha(q(j)+1,j).EQ.0.0)THEN
262         Rint = x1(q(j)+1)
          END IF
```

```
264     spR = spR + (2.0/3.0)*pie*(Rint**3.0)*sin(x2(j))*dx2
      END DO
266   Rint = (0.75*spR/pie)**(1.0/3.0)
      RETURN
268   END
```

NLAABC.f

```
      SUBROUTINE NLAABC (mfl)
 2
      INCLUDE "commonblock"
 4 C -----------------------------------------------------------------
   C AUTHOR: J. R. C. KING
 6 C ----------------------
   C CHANGE RECORD:
 8 C   20-07-2013: CREATED
   C   06-08-2013: MODIFIED TO UPSTREAM WENO-Z SCHEME
10 C   30-09-2013: FOR A GENERAL NODE 'ibn' AT RIGHT BOUND
   C   ??<12-2014: A BAZILLIION CHANGES. NOW 2D.
12 C -----------------------------------------------------------------
   C THIS SUBROUTINE USES THE STANDARD CHARACTERISTIC BOUNDARY COND-
14 C ITION FORMULATION PRESENTED IN Thompson, JCP 89, 439-461(1990)
   C TO APPLY BOUNDARY CONDITIONS AT THE RIGHT EDGE OF THE DOMAIN
16 C
   C TAKES A CELL INDEX ibn AND USES SOLUTION AT THIS CELL TO APPLY
18 C NLAA BC BY PROVIDING NEW VALUES FOR THIS CELL -----------------
   C -----------------------------------------------------------------
20    INTEGER mfl
      DOUBLE PRECISION drhodrB,durdrB,dutdrB,dpdrB
22    DOUBLE PRECISION drhodthB,durdthB,dutdthB,dpdthB
      DOUBLE PRECISION cB,L1B,L2B,L3B,L4B
24    DOUBLE PRECISION henth
      DOUBLE PRECISION aL1,NA,NB
26    DOUBLE PRECISION durdtB,dutdtB,drhodtB,dpdtB
      DOUBLE PRECISION newrho(nx2max),newp(nx2max),
28       +       newur(nx2max),newut(nx2max)
      DOUBLE PRECISION rob,robm1,robm2,robm3,robm,robp
30    DOUBLE PRECISION ub,ubm1,ubm2,ubm3,ubp,ubm
      DOUBLE PRECISION vb,vbm1,vbm2,vbm3,vbp,vbm
32    DOUBLE PRECISION pb,pbm1,pbm2,pbm3,pbm,pbp
      IF (mfl .EQ. 2) THEN
34 C FOR EVERY NODE ON THE nx1 BOUNDARY
      DO j=2,nx2-1
36      rob = Uprop(mfl,1,nx1,j)
        robm1 = Uprop(mfl,1,nx1-1,j)
38      robm2 = Uprop(mfl,1,nx1-2,j)
        robm3 = Uprop(mfl,1,nx1-3,j)
40      robm = Uprop(mfl,1,nx1,j-1)
        robp = Uprop(mfl,1,nx1,j+1)
42      ub = Uprop(mfl,2,nx1,j)/Uprop(mfl,1,nx1,j)
        ubm1 = Uprop(mfl,2,nx1-1,j)/Uprop(mfl,1,nx1-1,j)
44      ubm2 = Uprop(mfl,2,nx1-2,j)/Uprop(mfl,1,nx1-2,j)
        ubm3 = Uprop(mfl,2,nx1-3,j)/Uprop(mfl,1,nx1-3,j)
46      ubm = Uprop(mfl,2,nx1,j-1)/Uprop(mfl,1,nx1,j-1)
        ubp = Uprop(mfl,2,nx1,j+1)/Uprop(mfl,1,nx1,j+1)
48      vb = Uprop(mfl,3,nx1,j)/Uprop(mfl,1,nx1,j)
        vbm1 = Uprop(mfl,3,nx1-1,j)/Uprop(mfl,1,nx1-1,j)
50      vbm2 = Uprop(mfl,3,nx1-2,j)/Uprop(mfl,1,nx1-2,j)
        vbm3 = Uprop(mfl,3,nx1-3,j)/Uprop(mfl,1,nx1-3,j)
52      vbm = Uprop(mfl,3,nx1,j-1)/Uprop(mfl,1,nx1,j-1)
        vbp = Uprop(mfl,3,nx1,j+1)/Uprop(mfl,1,nx1,j+1)
54      pb = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1,j)-
         +       0.5*rob*(ub**2+vb**2))-gm(mfl)*pc(mfl)
56      pbm1 = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1-1,j)-
         +       0.5*robm1*(ubm1**2+vbm1**2))-gm(mfl)*pc(mfl)
58      pbm2 = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1-2,j)-
```

```
     +      0.5*robm2*(ubm2**2+vbm2**2))-gm(mfl)*pc(mfl)
     pbm3 = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1-3,j)-
     +      0.5*robm3*(ubm3**2+vbm3**2))-gm(mfl)*pc(mfl)
     pbm = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1,j-1)-
     +      0.5*robm*(ubm**2+vbm**2))-gm(mfl)*pc(mfl)
     pbp = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1,j+1)-
     +      0.5*robp*(ubp**2+vbp**2))-gm(mfl)*pc(mfl)
C    CALCULATE THE LOCAL SPEED OF SOUND --------------------
     cB = sqrt(gm(2)*(pb+pc(2))/rob)
C    FIRST CALCULATE DERIVATIVES OF PRIMITIVE VARIABLES AT -
C    BOUNDARY: 1 SIDED 3rd ORDER ---------------------------
     drhodrB = (-11.0*rob+18.0*robm1-9.0*robm2+2.0*robm3)/
     +      (-6.0*dx1)
     durdrB = (-11.0*ub+18.0*ubm1-9.0*ubm2+2.0*ubm3)/
     +      (-6.0*dx1)
     dutdrB = (-11.0*vb+18.0*vbm1-9.0*vbm2+2.0*vbm3)/
     +      (-6.0*dx1)
     dpdrB = (-11.0*pb+18.0*pbm1-9.0*pbm2+2.0*pbm3)/
     +      (-6.0*dx1)
C    DERIVATIVES IN THE THETA DIRECTION!!
     drhodthB= (robp-robm)/(2.0*dx2)
     durdthB = (ubp-ubm)/(2.0*dx2)
     dutdthB = (vbp-vbm)/(2.0*dx2)
     dpdthB = (pbp-pbm)/(2.0*dx2)
C    ASSUME L4 OUTGOING -------------------------------------
     L4B = (ub+cB)*(dpdrB + rob*cB*durdrB)
C    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C    NLAA BCs!!!
C    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
     henth = (pb-Pinf(j))/rob
C
     aL1 = (rob*(ub-cB)/x1(nx1))*((3.0-ub/cB)*0.5*ub**2.0
     +          + 2.0*ub*cB - (1.0+ub/cB)*henth)
     NA = 1.0 - 0.5*(ub-cB)*ub/(cB**2.0)
     NB = (1.0/(2.0*rob*cB))*(L4B-aL1)
     +      + vb*durdthB/x1(nx1)+ (vb**2.0)/x1(nx1)
     +      +grav*cos(x2(j))
     durdtB = -1.0*NB/NA
     L1B = aL1+rob*ub*(ub-cB)*durdtB/cB
     IF (ub .LT. 0.0) THEN
C      INCOMING
       L2B = ((rob*ub**2.0)/x1(nx1))*(henth/cB -
     +        2.0*ub + 0.5*(ub**2.0)/cB) + rob*ub*
     +        (1.0-ub/cB)*durdtB
       L3B = 0.0!ub*(-2.0)*vb/x1(nx1)
     ELSE
C      OUTGOING -----------------------------------
       L2B = ub*(cB*cB*drhodrB-dpdrB)
       L3B = ub*dutdrB
     END IF
C    DETERMINE THE TIME DERIVATIVES ------------------------
C      NEED TO CHECK SOURCE TERMS!!!
     drhodtB = -1.0*(L2B + 0.5*(L4B+L1B))/cB**2.0
     +      - vb*drhodthB/x1(nx1)
     +      - rob*dutdthB/x1(nx1)
     +      - 2.0*rob*ub/x1(nx1)
     +      - rob*vb/(x1(nx1)*tan(x2(j)))

     dpdtB = -0.5*(L4B+L1B)
     +      -vb*dpdthB/x1(nx1)
     +      -rob*cB*cB*dutdthB/x1(nx1)
     +      - 2.0*rob*ub*cB*cB/x1(nx1)
     +      - rob*vb*cB*cB/(x1(nx1)*tan(x2(j)))

     dutdtB = -1.0*L3B
     +      - vb*dutdthB/x1(nx1)
```

```
     +      - dpdthB/(rob*x1(nx1))
     +      - ub*vb/x1(nx1)
     +      + grav*sin(x2(j))
C   1st ORDER TIME INTEGRATION ----------------------------
    newrho(j) = rob + drhodtB*dt
    newp(j) = pb + dpdtB*dt
    newur(j) = ub + durdtB*dt
    newut(j) = vb + dutdtB*dt
  END DO
  DO j=2,nx2-1
    Uprop(mfl,1,nx1,j) = newrho(j)
    Uprop(mfl,2,nx1,j) = newrho(j)*newur(j)
    Uprop(mfl,3,nx1,j) = newrho(j)*newut(j)
    Uprop(mfl,4,nx1,j) = (newp(j)+gm(2)*pc(2))/(gm(2)-1.0)+
     +      0.5*newrho(j)*(newur(j)**2.0 + newut(j)**2.0)
  END DO
  ELSE
    DO j=2,nx2-1
C     STICK IN A REFLECTING BOUNDARY...
      Uprop(mfl,1,nx1,j) = Uprop(mfl,1,nx1-1,j)
      Uprop(mfl,2,nx1,j) = -1.0*Uprop(mfl,2,nx1-1,j)
      Uprop(mfl,3,nx1,j) = Uprop(mfl,3,nx1-1,j)
      Uprop(mfl,4,nx1,j) = Uprop(mfl,4,nx1-1,j)
    END DO
  END IF
  RETURN
  END
```

NLAABCGHOST.f

```
  SUBROUTINE NLAABCGHOST (mfl)

  INCLUDE "commonblock"
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   20-07-2013: CREATED
C   06-08-2013: MODIFIED TO UPSTREAM WENO-Z SCHEME
C   30-09-2013: FOR A GENERAL NODE 'ibn' AT RIGHT BOUND
C   ??<06-2014: NOW 2D...
c   25-06-2014: INCLUSION OF THE GHOST!!
C ---------------------------------------------------------------
C THIS SUBROUTINE USES THE STANDARD CHARACTERISTIC BOUNDARY COND-
C ITION FORMULATION PRESENTED IN Thompson, JCP 89, 439-461(1990)
C TO APPLY BOUNDARY CONDITIONS AT THE RIGHT EDGE OF THE DOMAIN
C
C TAKES A CELL INDEX ibn AND USES SOLUTION AT THIS CELL TO APPLY
C NLAA BC BY PROVIDING NEW VALUES FOR THIS CELL ------------------
C ---------------------------------------------------------------
C WARNING: The inclusion of the ghost will not work correctly if
C KE2D is run from a dump file. For simulations including the
C ghost, KE2D must be run from scratch.
C ---------------------------------------------------------------
  INTEGER mfl
  DOUBLE PRECISION drhodrB,durdrB,dutdrB,dpdrB
  DOUBLE PRECISION drhodthB,durdthB,dutdthB,dpdthB
  DOUBLE PRECISION cB,L1B,L2B,L3B,L4B
  DOUBLE PRECISION henth
  DOUBLE PRECISION aL1,NA,NB
  DOUBLE PRECISION durdtB,dutdtB,drhodtB,dpdtB
  DOUBLE PRECISION newrho(nx2max),newp(nx2max),
     +      newur(nx2max),newut(nx2max)
c o=o=o=o=o=o=o=o=o=o=o=o=o
  INTEGER ig
  DOUBLE PRECISION rg,thg,d,taug,ttarget,ref_ghost
```

```
37    DOUBLE PRECISION ug,pg,dugdrg,dpgdrg,rhog
      DOUBLE PRECISION ugr,ugth,dugrdr,dugrdth,dpgdr,dpgdth,dugthdr
39       +    ,dugthdth
      DOUBLE PRECISION cos1,cos2,sin1,sin2,gL1,gL2,gL3
41    DOUBLE PRECISION dubdt,dhdt
      IF (mfl .EQ. 2)THEN !if looking at water...
43  c 0. Set f,df,d2f...
      IF (tn .EQ. 1) THEN
45      ubold = ux1(nx1,2)
      END IF
47    cB = sqrt(gm(2)*(p(nx1,2)+pc(2))/rho(nx1,2))
      henth = (p(nx1,2)-pinf(2))/rho(nx1,2)
49    ug = ux1(nx1,2)
      dubdt = (ug-ubold)/dt
51    f(tn) = x1(nx1)*x1(nx1)*(ug-henth/cB -
         +       (ug**2)/(2.0*cB))
53    df(tn) = x1(nx1)*(henth + 0.5*ug**2)
      d2f(tn) = -1.0*cb*henth -0.5*cb*ug**2 + x1(nx1)*cB*dubdt
55    thist(tn) = t
      ubold = ux1(nx1,2)
57    d = dgun
    c o=o=o=o=o=o=o=o=o=o=o=o=o
59  C FOR EVERY NODE ON THE nx1 BOUNDARY
      DO j=2,nx2-1
61      rob = Uprop(mfl,1,nx1,j)
        robm1 = Uprop(mfl,1,nx1-1,j)
63      robm2 = Uprop(mfl,1,nx1-2,j)
        robm3 = Uprop(mfl,1,nx1-3,j)
65      robm = Uprop(mfl,1,nx1,j-1)
        robp = Uprop(mfl,1,nx1,j+1)
67      ub = Uprop(mfl,2,nx1,j)/Uprop(mfl,1,nx1,j)
        ubm1 = Uprop(mfl,2,nx1-1,j)/Uprop(mfl,1,nx1-1,j)
69      ubm2 = Uprop(mfl,2,nx1-2,j)/Uprop(mfl,1,nx1-2,j)
        ubm3 = Uprop(mfl,2,nx1-3,j)/Uprop(mfl,1,nx1-3,j)
71      ubm = Uprop(mfl,2,nx1,j-1)/Uprop(mfl,1,nx1,j-1)
        ubp = Uprop(mfl,2,nx1,j+1)/Uprop(mfl,1,nx1,j+1)
73      vb = Uprop(mfl,3,nx1,j)/Uprop(mfl,1,nx1,j)
        vbm1 = Uprop(mfl,3,nx1-1,j)/Uprop(mfl,1,nx1-1,j)
75      vbm2 = Uprop(mfl,3,nx1-2,j)/Uprop(mfl,1,nx1-2,j)
        vbm3 = Uprop(mfl,3,nx1-3,j)/Uprop(mfl,1,nx1-3,j)
77      vbm = Uprop(mfl,3,nx1,j-1)/Uprop(mfl,1,nx1,j-1)
        vbp = Uprop(mfl,3,nx1,j+1)/Uprop(mfl,1,nx1,j+1)
79      pb = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1,j)-
         +       0.5*rob*(ub**2+vb**2))-gm(mfl)*pc(mfl)
81      pbm1 = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1-1,j)-
         +       0.5*robm1*(ubm1**2+vbm1**2))-gm(mfl)*pc(mfl)
83      pbm2 = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1-2,j)-
         +       0.5*robm2*(ubm2**2+vbm2**2))-gm(mfl)*pc(mfl)
85      pbm3 = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1-3,j)-
         +       0.5*robm3*(ubm3**2+vbm3**2))-gm(mfl)*pc(mfl)
87      pbm = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1,j-1)-
         +       0.5*robm*(ubm**2+vbm**2))-gm(mfl)*pc(mfl)
89      pbp = (gm(mfl)-1.0)*(Uprop(mfl,4,nx1,j+1)-
         +       0.5*robp*(ubp**2+vbp**2))-gm(mfl)*pc(mfl)
91  C   CALCULATE THE LOCAL SPEED OF SOUND --------------------
        cB = sqrt(gm(2)*(pb+pc(2))/rob)
93  C   FIRST CALCULATE DERIVATIVES OF PRIMITIVE VARIABLES AT -
    C   BOUNDARY: 1 SIDED 3rd ORDER ---------------------------
95      drhodrB = (-11.0*rob+18.0*robm1-9.0*robm2+2.0*robm3)/
         +       (-6.0*dx1)
97      durdrB = (-11.0*ub+18.0*ubm1-9.0*ubm2+2.0*ubm3)/
         +       (-6.0*dx1)
99      dutdrB = (-11.0*vb+18.0*vbm1-9.0*vbm2+2.0*vbm3)/
         +       (-6.0*dx1)
101     dpdrB = (-11.0*pb+18.0*pbm1-9.0*pbm2+2.0*pbm3)/
         +       (-6.0*dx1)
```

```
C   DERIVATIVES IN THE THETA DIRECTION!!
    drhodthB= (robp-robm)/(2.0*dx2)
    durdthB = (ubp-ubm)/(2.0*dx2)
    dutdthB = (vbp-vbm)/(2.0*dx2)
    dpdthB = (pbp-pbm)/(2.0*dx2)
C   ASSUME L4 OUTGOING ------------------------------------
    L4B = (ub+cB)*(dpdrB + rob*cB*durdrB)
c   =o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=
c   GHOST!!
c   =<o>=<o>=<o>=<o>=<o>=<o>=<o>=<o>=<o>=<o>=<o>=<o>=<o>=<o
c   1. Find rg,thg
    thg = atan(sin(x2(j))/((2.0*d/x1(nx1))-cos(x2(j))))
    rg = x1(nx1)*sin(x2(j))/sin(thg)
c   2. Caclulate delay...
    taug = (rg-x1(nx1))/cB
c   search through to find the index of f we're looking for
    ttarget = t-taug
    IF (ttarget .LE. 0)THEN
      ig = 1
    ELSE
      DO i=1,tn
        IF (thist(i).LE.ttarget.AND.
     +        thist(i+1).GT.ttarget)THEN
          ig=i
        END IF
      END DO
    END IF
c   3. set ug,pg,dugdrg,dpgdrg,rhog
    rhog=rho2 !CHEAT!
    IF (ghostflag .EQ. 1)THEN
      ref_ghost = 1.0
    ELSE IF (ghostflag .EQ. -1)THEN
      ref_ghost = -1.0
    ELSE
      ref_ghost = 0.0
    END IF
    ug = ref_ghost*(f(ig)/(rg*rg) + df(ig)/(rg*cb))
    pg = ref_ghost*(df(ig)/rg - 0.5*ug*ug)*rhog + pinf(j)
    dugdrg = ref_ghost*(-2.0*f(ig)/(rg**3) -
     +      2.0*df(ig)/(rg*rg*cb) - d2f(ig)/(rg*cb*cb))
    dpgdrg = -1.0*ref_ghost*rhog*(ug*dugdrg +df(ig)/(rg*rg)
     +       + d2f(ig)/(rg*cb))
c   4. Find ugr,ugth,dugrdr,dugrdth,dpgdr,dpgdth,
c   dugthdr,dugthdth
    cos1 = cos(pie-x2(j)-thg)
    sin1 = sin(pie-x2(j)-thg)
    ugr = ug*cos1
    ugth = ug*sin1
    dugrdr = dugdrg*cos1*cos1
    dugrdth = x1(nx1)*dugdrg*sin1*cos1 + ug*sin1
    dugthdr = dugdrg*cos1*sin1
    dugthdth = x1(nx1)*dugdrg*sin1*sin1 - ug*cos1
    dpgdr = dpgdrg*cos1
    dpgdth = x1(nx1)*dpgdrg*sin1
  !!!!!!  Modulate the hydrostatic pressure!!
    pinf(j)=pinf0(j)+ref_ghost*(df(ig)/rg-0.5*ug*ug)*rhog
c   5. Use the above to set gL1,gL2,gL3
    IF (ttarget .LE. 0.0D0) THEN
      gL1 = 0.0D0
      gL2 = 0.0D0
      gL3 = 0.0D0
    ELSE
      gL1 = (ux1(nx1,j)-cB)*(dpgdr-rhog*cb*dugrdr)
      gL2 = -1.0*ux1(nx1,j)*dpgdr
      gL3 = ux1(nx1,j)*dugthdr
    END IF
```

```
C   NLAA BCs!!!
C   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    henth = (pb-Pinf(j))/rob
C
    aL1 = (rob*(ub-cB)/x1(nx1))*((3.0-ub/cB)*0.5*ub**2.0
     +          + 2.0*ub*cB - (1.0+ub/cB)*henth)
     +      + gL1 !!!!!! =o=o=o=o=
    NA = 1.0 - 0.5*(ub-cB)*ub/(cB**2.0)
    NB = (1.0/(2.0*rob*cB))*(L4B-aL1)
     +      + vb*durdthB/x1(nx1)+ (vb**2.0)/x1(nx1)
     +      +grav*cos(x2(j)) !
    durdtB = -1.0*NB/NA
    L1B = aL1+rob*ub*(ub-cB)*durdtB/cB
    IF (ub .LT. 0.0) THEN
C     INCOMING
      L2B = ((rob*ub**2.0)/x1(nx1))*(henth/cB -
     +        2.0*ub + 0.5*(ub**2.0)/cB) + rob*ub*
     +        (1.0-ub/cB)*durdtB
     +        + gL2 !!! =o=o=o=o=o=
      L3B = 0.0 + gL3 !!=o=o=o=o=o=o
    ELSE
C     OUTGOING -------------------------------------
      L2B = ub*(cB*cB*drhodrB-dpdrB)
      L3B = ub*dutdrB
    END IF
C   DETERMINE THE TIME DERIVATIVES -----------------------
C     NEED TO CHECK SOURCE TERMS!!!
    drhodtB = -1.0*(L2B + 0.5*(L4B+L1B))/cB**2.0
     +      - vb*drhodthB/x1(nx1)
     +      - rob*dutdthB/x1(nx1)
     +      - 2.0*rob*ub/x1(nx1)
     +      - rob*vb/(x1(nx1)*tan(x2(j)))

    dpdtB = -0.5*(L4B+L1B)
     +      -vb*dpdthB/x1(nx1)
     +      -rob*cB*cB*dutdthB/x1(nx1)
     +      - 2.0*rob*ub*cB*cB/x1(nx1)
     +      - rob*vb*cB*cB/(x1(nx1)*tan(x2(j)))

    dutdtB = -1.0*L3B
     +      - vb*dutdthB/x1(nx1)
     +      - dpdthB/(rob*x1(nx1))
     +      - ub*vb/x1(nx1)
     +      + grav*sin(x2(j))
C   1st ORDER TIME INTEGRATION ---------------------------
    newrho(j) = rob + drhodtB*dt
    newp(j) = pb + dpdtB*dt
    newur(j) = ub + durdtB*dt
    newut(j) = vb + dutdtB*dt
  END DO
  DO j=2,nx2-1
    Uprop(mfl,1,nx1,j) = newrho(j)
    Uprop(mfl,2,nx1,j) = newrho(j)*newur(j)
    Uprop(mfl,3,nx1,j) = newrho(j)*newut(j)
    Uprop(mfl,4,nx1,j) = (newp(j)+gm(2)*pc(2))/(gm(2)-1.0)+
     +      0.5*newrho(j)*(newur(j)**2.0 + newut(j)**2.0)
  END DO
  ELSE
    DO j=2,nx2-1
C     STICK IN A REFLECTING BOUNDARY...
      Uprop(mfl,1,nx1,j) = Uprop(mfl,1,nx1-1,j)
      Uprop(mfl,2,nx1,j) = -1.0*Uprop(mfl,2,nx1-1,j)
      Uprop(mfl,3,nx1,j) = Uprop(mfl,3,nx1-1,j)
      Uprop(mfl,4,nx1,j) = Uprop(mfl,4,nx1-1,j)
    END DO
  END IF
```

```
235   RETURN
      END
```

RIEMANNHLLC.f

```
      SUBROUTINE RIEMANNHLLC(rhoL,momL,momTL,ENL,rhoR,momR,momTR,ENR,
     +           GAMR,PCRITR,
     +           fl1,fl2,fl3,fl4)

      IMPLICIT NONE
C -------------------------------------------------------------
C SUBROUTINE TO SOLVE A RIEMANN PROBLEM - HLLC SOLVER
C -------------------------------------------------------------
C Returns the fluxes directly
C -------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   19-03-2013: CREATED
C   28-05-2013: MODIFIED TO RETURN FLUXES
C   29-07-2013: MODIFIED TO DEAL WITH ONLY 1 FLUID
C   07-08-2013: MODIFIED TO TAKE CONS VARS FOR INPUT
C   ??<06-2014: INCLUSION OF TRANSVERSE VELOCITY
C -------------------------------------------------------------
      DOUBLE PRECISION  rhoL,momL,ENL,rhoR,momR,ENR,
     +        GAMR,PCRITR,delA,momTL,momTR
C
      DOUBLE PRECISION  uL,PL,uR,PR,utL,utR
      DOUBLE PRECISION  pstar,ustar,rhoLstar,rhoRstar
      DOUBLE PRECISION  ENLstar,ENRstar,eL,eR,HL,HR
      DOUBLE PRECISION  cL,cR
      DOUBLE PRECISION  bL,bR,bM
      DOUBLE PRECISION  ubar,cbar,rhobar,Povrhobar,Pbar
      DOUBLE PRECISION  hbar
      DOUBLE PRECISION  alf1,alf2,alf3
C FLUXES!!
      DOUBLE PRECISION  fl1,fl2,fl3,fl4
      LOGICAL     subsonicflag
C CALCULATE PRIMITIVE VARIABLES FROM CONSERVATIVE VARIABLES
      uL = momL/rhoL
      uR = momR/rhoR
      utL = momTL/rhoL
      utR = momTR/rhoR
      PL = (GAMR-1.0)*(ENL-0.5*rhoL*(uL**2.0 + utL**2)) - GAMR*PCRITR
      PR = (GAMR-1.0)*(ENR-0.5*rhoR*(uR**2.0 + utR**2)) - GAMR*PCRITR
c CALCULATE LEFT AND RIGHT SPEEDS OF SOUND ======================
      cL = sqrt(GAMR*(PL+PCRITR)/rhoL)
      cR = sqrt(GAMR*(PR+PCRITR)/rhoR)
C LITTLE ENERGY AND ENTHALPY
      eL = (PL+GAMR*PCRITR)/((GAMR-1.0)*rhoL)
      eR = (PR+GAMR*PCRITR)/((GAMR-1.0)*rhoR)
      HL = eL + PL/rhoL + 0.5*uL**2.0
      HR = eR + PR/rhoR + 0.5*uR**2.0
C CALCULATE ROE AVERAGES ((.)bar ) FOR DENSITY AND PRESSURE =====
      rhobar = sqrt(rhoL*rhoR)
      call ROEAVG (rhoL,rhoR,uL,uR,ubar)
      call ROEAVG (rhoL,rhoR,HL,HR,hbar)
      Povrhobar = (PL/sqrt(rhoL) + PR/sqrt(rhoR))/
     +      (sqrt(rhoL)+sqrt(rhoR)) +
     +      0.5*((uR-uL)/(sqrt(rhoL)+sqrt(rhoR)))**2.0
      Pbar = rhobar*Povrhobar
C JUST ONE FLUID ------------------------------------------------
C cbar = sqrt(GAMR*(Pbar + PCRITR)/rhobar)
C cbar = sqrt(GAMR*Povrhobar + PCRITR/rhobar)
      cbar = sqrt((GAMR-1.0)*(hbar-0.5*ubar**2))
C CALCULATE STAR STATES AND WAVE SPEEDS ========================
```

```
      bL = min(uL-cL,ubar-cbar)
      bR = max(ubar+cbar,uR+cR)
      ustar = (rhoR*uR*(bR-uR) - rhoL*uL*(bL-uL) + PL - PR)/
     +    (rhoR*(bR-uR) - rhoL*(bL-uL))
      bM = ustar
      rhoLstar = rhoL*(bL-uL)/(bL-ustar)
      rhoRstar = rhoR*(bR-uR)/(bR-ustar)
      pstar = PL + rhoL*(uL-bL)*(uL-ustar)
      ENLstar = (pstar + GAMR*PCRITR)/(GAMR-1.0) +
     +    0.5*rhoLstar*ustar**2.0
      pstar = PR + rhoR*(bR-uR)*(ustar-uR)
      ENRstar = (pstar + GAMR*PCRITR)/(GAMR-1.0) +
     +    0.5*rhoRstar*ustar**2.0
C     CALCULATE FLUXES DIRECTLY!!
      subsonicflag = .TRUE.
      IF (bL .GT. 0.0) THEN
        fl1 = rhoL*uL
        fl2 = fl1*uL + PL
        fl3 = fl1*utL
        fl4 = uL*(PL + ENL)
        subsonicflag = .FALSE.
      END IF
      IF (bR .LT. 0.0) THEN
        fl1 = rhoR*uR
        fl2 = fl1*uR + PR
        fl3 = fl1*utR
        fl4 = uR*(PR + ENR)
        subsonicflag = .FALSE.
      END IF
      IF (subsonicflag) THEN
        IF (bM .GT. 0.0) THEN
          fl1 = rhoL*uL + bL*(rhoLstar - rhoL)
          fl2 = rhoL*uL*uL +
     +          bL*(rhoLstar*ustar - rhoL*uL) + PL
          fl3 = rhoL*uL*utL
          fl4 = uL*(PL+ENL) + bL*(ENLstar - ENL)
        ELSE
          fl1 = rhoR*uR + bR*(rhoRstar - rhoR)
          fl2 = rhoR*uR*uR +
     +          bR*(rhoRstar*ustar - rhoR*uR) + PR
          fl3 = rhoR*uR*utR
          fl4 = uR*(PR+ENR) + bR*(ENRstar - ENR)
        END IF
      END IF
      RETURN
      END
C -------------------------------------------------------------
      SUBROUTINE ROEAVG (roL,roR,ftnL,ftnR,RAVG)

      IMPLICIT NONE
      DOUBLE PRECISION roL,roR,ftnL,ftnR,RAVG
      RAVG = (sqrt(roL)*ftnL + sqrt(roR)*ftnR)/
     +      (sqrt(roL)+sqrt(roR))
      RETURN
      END
C -------------------------------------------------------------
      SUBROUTINE RIEMANNHLLCVA (rhoL,momL,ENL,rhoR,momR,ENR,
     +          GAMR,PCRITR,delA,
     +          fl1,fl2,fl3)

      IMPLICIT NONE
C -------------------------------------------------------------
C SUBROUTINE TO SOLVE A RIEMANN PROBLEM - HLLC SOLVER - WITH
C FLOW AREA VARIATION COURTESY OF ROE!
C -------------------------------------------------------------
C Returns the fluxes directly
```

```
C -------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   19-03-2013: CREATED
C   28-05-2013: MODIFIED TO RETURN FLUXES
C   29-07-2013: MODIFIED TO DEAL WITH ONLY 1 FLUID
C   07-08-2013: MODIFIED TO TAKE CONS VARS FOR INPUT
C   08-10-2013: RE-CREATED. AREA VARIATION.
C -------------------------------------------------------------
      DOUBLE PRECISION  rhoL,momL,ENL,rhoR,momR,ENR,
     +        GAMR,PCRITR,delA
C
      DOUBLE PRECISION  uL,PL,uR,PR
      DOUBLE PRECISION  pstar,ustar,rhoLstar,rhoRstar
      DOUBLE PRECISION  ENLstar,ENRstar,eL,eR,HL,HR
      DOUBLE PRECISION  cL,cR
      DOUBLE PRECISION  bL,bR,bM
      DOUBLE PRECISION  ubar,cbar,rhobar,Povrhobar,Pbar
      DOUBLE PRECISION  hbar
      DOUBLE PRECISION  alf1,alf2,alf3
C FLUXES!!
      DOUBLE PRECISION  fl1,fl2,fl3
      LOGICAL     subsonicflag
C CALCULATE PRIMITIVE VARIABLES FROM CONSERVATIVE VARIABLES
      uL = momL/rhoL
      uR = momR/rhoR
      PL = (GAMR-1.0)*(ENL-0.5*rhoL*uL**2.0) - GAMR*PCRITR
      PR = (GAMR-1.0)*(ENR-0.5*rhoR*uR**2.0) - GAMR*PCRITR
c CALCULATE LEFT AND RIGHT SPEEDS OF SOUND ======================
      cL = sqrt(GAMR*(PL+PCRITR)/rhoL)
      cR = sqrt(GAMR*(PR+PCRITR)/rhoR)
C LITTLE ENERGY AND ENTHALPY
      eL = (PL+GAMR*PCRITR)/((GAMR-1.0)*rhoL)
      eR = (PR+GAMR*PCRITR)/((GAMR-1.0)*rhoR)
      HL = eL + PL/rhoL + 0.5*uL**2.0
      HR = eR + PR/rhoR + 0.5*uR**2.0
C CALCULATE ROE AVERAGES ((.)bar ) FOR DENSITY AND PRESSURE =====
      rhobar = sqrt(rhoL*rhoR)
      call ROEAVG (rhoL,rhoR,uL,uR,ubar)
      call ROEAVG (rhoL,rhoR,HL,HR,hbar)
      Povrhobar = (PL/sqrt(rhoL) + PR/sqrt(rhoR))/
     +      (sqrt(rhoL)+sqrt(rhoR)) +
     +      0.5*((uR-uL)/(sqrt(rhoL)+sqrt(rhoR)))**2.0
      Pbar = rhobar*Povrhobar
C JUST ONE FLUID ------------------------------------------------
C cbar = sqrt(GAMR*(Pbar + PCRITR)/rhobar)
C cbar = sqrt(GAMR*Povrhobar + PCRITR/rhobar)
      cbar = sqrt((GAMR-1.0)*(hbar-0.5*ubar**2))
C DO ROE SOLVER...
      alf1 = (0.5/(cbar**2.0))*((PR-PL)-rhobar*cbar*(uR-uL)+
     +    rhobar*ubar*cbar*cbar*delA/(ubar-cbar))
      alf2 = (1.0/(cbar**2.0))*(cbar*cbar*(rhoR-rhoL)-(PR-PL))
      alf3 = (0.5/(cbar**2.0))*((PR-PL)+rhobar*cbar*(uR-uL)+
     +    rhobar*ubar*cbar*cbar*delA/(ubar+cbar))
      fl1 = 0.5*(momL+momR)
     +    - 0.5*(alf1*abs(ubar-cbar)*(1.0)
     +    + alf2*abs(ubar)*(1.0)
     +    + alf3*abs(ubar+cbar)*(1.0))
      fl2 = 0.5*(momL*uL+momR*uR) + 0.5*(PL+PR)
     +    - 0.5*(alf1*abs(ubar-cbar)*(ubar-cbar)
     +    + alf2*abs(ubar)*(ubar)
     +    + alf3*abs(ubar+cbar)*(ubar+cbar))
      fl3 = 0.5*(uL*(ENL+PL)+uR*(ENR+PR))
     +    - 0.5*(alf1*abs(ubar-cbar)*(hbar-ubar*cbar)
     +    + alf2*abs(ubar)*(0.5*ubar*ubar)
```

```
     +    + alf3*abs(ubar+cbar)*(hbar+ubar*cbar))
  RETURN
  END
```

SETBCELL.f

```
  SUBROUTINE SETBCELL

  INCLUDE "commonblock"
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   04-03-2014: CREATED
C ---------------------------------------------------------------
C SET BCELL VALUES
C ---------------------------------------------------------------
C DECLARATIONS!
  INTEGER im1,ip1,jm1,jp1
  DOUBLE PRECISION spim,spip,spjm,spjp
  DOUBLE PRECISION spimjm,spimjp,spipjm,spipjp
  DOUBLE PRECISION storeA
C Set a flag to say whether a cell is beside the interface...
  bcount = 0
  DO i=2,nx1-1
    DO j=1,nx2
      im1=max(1,i-1)
      ip1=min(nx1,i+1)
      jm1=max(1,j-1)
      jp1=min(nx2,j+1)
      spim = alpha(i,j)*alpha(im1,j)
      spip = alpha(i,j)*alpha(ip1,j)
      spjm = alpha(i,j)*alpha(i,jm1)
      spjp = alpha(i,j)*alpha(i,jp1)
      spimjm = alpha(i,j)*alpha(im1,jm1)
      spimjp = alpha(i,j)*alpha(im1,jp1)
      spipjm = alpha(i,j)*alpha(ip1,jm1)
      spipjp = alpha(i,j)*alpha(ip1,jp1)
      storeA = min(spim,spip,spjm,spjp,spimjm,spimjp,
     +        spipjm,spipjp)
      IF (storeA.GT.0)THEN
        bcell(i,j) = 0
      ELSE
        bcount = bcount + 1
        bci(bcount) = i
        bcj(bcount) = j
        IF (alpha(i,j).LE.0) THEN
          bcell(i,j) = 1
        ELSE
          bcell(i,j) = 2
        END IF
      END IF
    END DO
  END DO
  RETURN
  END
```

SETTSTEP.f

```
  SUBROUTINE SETTSTEP

  INCLUDE "commonblock"
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
```

```
C CHANGE RECORD:
C   20-11-2012: CREATED
C ----------------------------------------------------------------
C SUBROUTINE TO CALCULATE THE TIME STEP BASED ON CFL NUMBER, GRID
C SIZE, MAXIMUM VELOCITY AND MAXIMUM SOUND SPEED
C ----------------------------------------------------------------
  DOUBLE PRECISION strx1,strx2,strmax
C FIND THE LARGEST SPEEDS ---------------------------------------
  strmax = 0.0
  DO i=1,nx1
    DO j=1,nx2
      IF (alpha(i,j) .LE. 0 ) THEN
c       Here there is a bodge - c = max(c,300).
        lc(i,j) = max(sqrt(gm(1)*(p(i,j)+Pc(1))
     +          /rho(i,j)),00.0)
       ELSE
        lc(i,j) = sqrt(gm(2)*(p(i,j)+Pc(2))
     +          /rho(i,j))
       END IF
       strx1 = (lc(i,j)+abs(ux1(i,j)))/dx1
       strx2 = (lc(i,j)+abs(ux2(i,j)))/(x1(i)*dx2)
       IF((strx1+strx2).GT.strmax) THEN
         strmax = strx1+strx2
c        write(6,*) "dt limit cell = ",i,j
       END IF
    END DO
  END DO
C SET THE TIME STEP ---------------------------------------------
  dt = CFL/strmax
c REMOVE THIS WHEN FINISHED DEBUGGING w.r.t. 1D code
c dt = 1e-5
  RETURN
  END
```

SETUP.f

```
  SUBROUTINE SETUP

  INCLUDE "commonblock"
C ----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   20-11-2012: CREATED
C   05-08-2013: A MYRIAD OF CHANGES, UNRECORDED
C ----------------------------------------------------------------
C SUBROUTINE TO READ THE INPUT FILES FOR PROGRAM KINGEULER1D
C AND TO SET THE INITIAL FIELDS. THIS SUBROUTINE IS A BIT MESSY.
C ----------------------------------------------------------------
  INTEGER dump_read_flag
  DOUBLE PRECISION storr2,eta
  READ (1,*) nx1
  READ (1,*) nx2
  READ (1,*) nt
  READ (1,*) outfreq
  READ (1,*) x1d
  READ (1,*) x2d
  READ (1,*) CFL
  READ (1,*) coordsno
  READ (1,*) grav
  READ (1,*) viscflag
  READ (1,*) ghostflag
  READ (1,*) dgun
  READ (1,*) dump_read_flag
  READ (1,*) boundflag
  READ (1,*) distflag
```

```
31    OPEN(unit=50,file="AG.dat")
      READ (50,*) rho1
33    READ (50,*) p1
      READ (50,*) ux11
35    READ (50,*) ux21
      READ (50,*) gm(1)
37    READ (50,*) pc(1)
      READ (50,*) rho2
39    READ (50,*) p2
      READ (50,*) ux12
41    READ (50,*) ux22
      READ (50,*) gm(2)
43    READ (50,*) pc(2)
      READ (50,*) Rint
45    IF (dgun .LE. 1D3) THEN
        p2=p2+(dgun-7.7D0)*rho2*9.81D0 !this is ghost stuff
47    ELSE  ! to make this work, for G/NG comparisons, set NG
        p2=p2+(1D-3*dgun-7.7D0)*rho2*9.81D0 !depth to G depth
49    END IF ! multiplied by 1000!
C SET THE INITIAL FIELDS ----------------------------------------
51  C CREATE THE MESH
      call MESHGEN
53    DO i=1,nx1
        DO j=1,nx2
55  C     SET THE FLOW PARAMETERS ACCORDING TO WHETHER WE'RE ----
    C     INSIDE OR OUTSIDE Rint -------------------------------
57  C     eta is a disturbance. Set to zero mostly...
          IF(distflag.EQ.0)THEN
59          eta = 0.0
          ELSE IF (distflag.EQ.1)THEN
61          eta = 1D-3*sin(20*x2(j))
          ELSE IF (distflag.EQ.2)THEN
63          eta = 1D-3*cos(20*x2(j))
          END IF
65        IF (x1(i).LE. Rint+eta) THEN
            rho(i,j) = rho1
67          P(i,j) = p1 - rho(i,j)*grav*z(i,j)
            ux1(i,j) = ux11
69          ux2(i,j) = ux21
            E(i,j)=(P(i,j)+Pc(1)*gm(1))/(gm(1)-1)+
71       +         0.5D0*rho(i,j)*(ux1(i,j)**2+
         +         ux2(i,j)**2)
73        ELSE
            rho(i,j) = rho2
75          p(i,j) = p2 - rho(i,j)*grav*z(i,j)
            ux1(i,j) = ux12
77          ux2(i,j) = ux22
            E(i,j)=(P(i,j)+Pc(2)*gm(2))/(gm(2)-1)+
79       +         0.5D0*rho(i,j)*(ux1(i,j)**2+
         +         ux2(i,j)**2)
81        END IF
          alpha(i,j) = x1(i) - Rint - eta
83        ulsx1(i,j) = ux1(i,j)
          ulsx2(i,j) = ux2(i,j)
85      END DO
      END DO
87    DO j=1,nx2
        Pinf(j) = P(nx1,j)
89      Pinf0(j) = P(nx1,j)
      END DO
91  C Counter for outputting
      ofc1 = outfreq + 1
93    t=0.0
    C Loading from dump?
95  c Shall we start from the last output?! Go on then!!
      IF (dump_read_flag.eq.1)THEN
```

```
97      OPEN(3,file="DUMP.tron")
        READ(3,*) t
99      DO i=1,nx1
          DO j=1,nx2
101         READ (3,*) rho(i,j),p(i,j),ux1(i,j),
         +          ux2(i,j),E(i,j),alpha(i,j)
103         ulsx1(i,j) = ux1(i,j)
            ulsx2(i,j) = ux2(i,j)
105       END DO
        END DO
107   END IF
      RETURN
109   END
C -------------------------------------------------------------
111   SUBROUTINE MESHGEN

113   INCLUDE "commonblock"
C -------------------------------------------------------------
115 C CREATE THE MESH BASED ON x1d,x2d,nx1,nx2. DETERMINE CELL FACE
    C AREAS AND CELL VOLUMES.
117 C -------------------------------------------------------------
      DOUBLE PRECISION x1min,x2min,x1ph,x1mh,x2ph,x2mh
119   x1min = 0.0D0
      x2min = 0.0D0
121 !!  dx1 = 1.0D0/49.0D0 !use this to compare domain sizes
      dx1 = (x1d-x1min)/float(nx1-1) !use this normally
123   dx2 = (x2d-x2min)/float(nx2-2)
      DO i=1,nx1
125     x1(i) = x1min + dx1*float(i-1) + 0.5D0*dx1
      END DO
127   DO j=1,nx2
        x2(j) = x2min + dx2*float(j-1) - 0.5D0*dx2
129   END DO
C CALCULATE x,z,r,th VALUES FOR EACH NODE, FOR OUTPUTTING...
131   IF (coordsno.EQ.2.0) THEN
        DO i=1,nx1
133       DO j=1,nx2
            x(i,j) = x1(i)*sin(x2(j))
135         z(i,j) = x1(i)*cos(x2(j))
            r(i,j) = x1(i)
137         th(i,j) = x2(j)
          END DO
139     END DO
      ELSE IF (coordsno.EQ.1.0)THEN
141     DO i=1,nx1
          DO j=1,nx2
143         x(i,j) = x1(i)
            z(i,j) = x2(j)
145         r(i,j) = sqrt(x1(i)**2 + x2(j)**2)
            th(i,j) = acos(x2(j)/r(i,j))
147       END DO
        END DO
149   ELSE
        DO i=1,nx1
151       DO j=1,nx2
            x(i,j) = x1(i)
153         z(i,j) = x2(j)
            r(i,j) = sqrt(x1(i)**2 + x2(j)**2)
155         th(i,j) = acos(x2(j)/r(i,j))
          END DO
157     END DO
      END IF
159   DO i=1,nx1
        DO j=1,nx2
161       WRITE(30,*) x1(i)
          WRITE(31,*) x2(j)
```

```
      WRITE(32,*) x(i,j)
      WRITE(33,*) z(i,j)
    END DO
  END DO
  CLOSE(30)
  CLOSE(31)
  CLOSE(32)
  CLOSE(33)
  RETURN
  END
```

SWEEP1D.f

```
  SUBROUTINE SWEEP1D (mfl,dflag)

  INCLUDE "commonblock"
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   20-07-2013: CREATED
C   07-08-2013: SPHERICALITY BY SOURCE TERMS
C   07-08-2013: MODIFIED TO WORK ON CONSERVATIVE VARS
C   13-08-2013: MUSCL SCHEME IS NOW 2-STEP(ie. CORRECT)
C   14-08-2013: STOPPED SOLVING DUPLICATE RIEMANN PROBS
C   14-08-2013: ONLY USE NECESSARY GHOST CELLS..
C   13-09-2013: MUSCL RECONSTRUCTION OF PRIMITIVE VARS
C   25-09-2013: WENO-Z RECONSTRUCTION OPTION!
C   25-09-2013: 3rd ORDER TVD TIME INTEGRATION OPTION!
C ---------------------------------------------------------------
C THE 1 PHASE EULER SOLVER. CHOOSES A FIELD TO WORK ON. RECONSTR-
C UCTS THE FIELD. CALCULATES FLUXES. UPDATES THE VALUES. ADDS THE
C SOURCE TERMS. UPDATES THE VALUES OF THE CHOSEN FIELD.
C ---------------------------------------------------------------
  INTEGER dflag,mfl
  INTEGER x1LOW,x1HIGH,x2LOW,x2HIGH
  DOUBLE PRECISION tsfrac,fiph(4,nx1max),fjph(4,nx2max)
  DOUBLE PRECISION sr,sux1,sux2,se,sp
  DOUBLE PRECISION S(4),cosoverRsin
  x1LOW = 1
  x1HIGH = nx1
  x2LOW = 1
  x2HIGH = nx2
  tsfrac = 1.0
  IF (dflag.EQ.1) THEN
C SWEEPS IN THE x1 DIRECTION
  DO j=x2LOW+1,x2HIGH-1
C   CALCULATE FLUXES
    DO i=x1LOW,x1HIGH-1
C     CALCULATE AT RIGHT "CELL WALLS": i,i+1
      call RIEMANNHLLC (Uprop(mfl,1,i,j),
     +      Uprop(mfl,2,i,j),Uprop(mfl,3,i,j),
     +      Uprop(mfl,4,i,j),Uprop(mfl,1,i+1,j),
     +      Uprop(mfl,2,i+1,j),Uprop(mfl,3,i+1,j),
     +      Uprop(mfl,4,i+1,j),
     +      gm(mfl),Pc(mfl),
     +      fiph(1,i),fiph(2,i),fiph(3,i),fiph(4,i))
    END DO
C   USE THE FLUXES TO UPDATE THE PROPERTIES ---------------
    DO i=x1LOW+1,x1HIGH-1
      sr = Uprop(mfl,1,i,j)
      sux1 = Uprop(mfl,2,i,j)/sr
      sux2 = Uprop(mfl,3,i,j)/sr
      se = Uprop(mfl,4,i,j)
      sp=(gm(mfl)-1)*(se-0.5*sr*(sux1*2 + sux2**2))-
     +        gm(mfl)*pc(mfl)
```

```
      S(1) = 2.0*sr*sux1/x1(i)
      S(2) = 2.0*sr*sux1*sux1/x1(i)
      S(3) = 2.0*sr*sux1*sux2/x1(i)
      S(4) = 2.0*sux1*(se+sp)/x1(i)
      DO k=1,4
        Uprop(mfl,k,i,j) = Uprop(mfl,k,i,j) -
     +        (tsfrac*dt/dx1)*
     +        (fiph(k,i)-fiph(k,i-1)) -
     +        tsfrac*dt*S(k)
      END DO
    END DO
  END DO
  ELSE IF (dflag.EQ.2) THEN
C SWEEPS IN THE x2 DIRECTION
  DO i=x1LOW+1,x1HIGH-1
C     CALCULATE FLUXES
    DO j=x2LOW,x2HIGH-1
C     CALCULATE AT RIGHT "CELL WALLS": j,j+1
      call RIEMANNHLLC (Uprop(mfl,1,i,j),
     +      Uprop(mfl,3,i,j),Uprop(mfl,2,i,j),
     +      Uprop(mfl,4,i,j),Uprop(mfl,1,i,j+1),
     +      Uprop(mfl,3,i,j+1),Uprop(mfl,2,i,j+1),
     +      Uprop(mfl,4,i,j+1),
     +      gm(mfl),Pc(mfl),
     +      fjph(1,j),fjph(3,j),fjph(2,j),fjph(4,j))
    END DO
C   USE THE FLUXES TO UPDATE THE PROPERTIES ---------------
    DO j=x2LOW+1,x2HIGH-1
      sr = Uprop(mfl,1,i,j)
      sux1 = Uprop(mfl,2,i,j)/sr
      sux2 = Uprop(mfl,3,i,j)/sr
      se = Uprop(mfl,4,i,j)
      sp=(gm(mfl)-1)*(se-0.5*sr*(sux1*2 + sux2**2))-
     +        gm(mfl)*pc(mfl)
      cosoverRsin = cos(x2(j))/(x1(i)*sin(x2(j)))
      S(1) = sr*sux2*cosoverRsin
      S(2) = sr*sux1*sux2*cosoverRsin
      S(3) = sr*sux2*sux2*cosoverRsin
      S(4) = sux2*(se+sp)*cosoverRsin
      DO k=1,4
        Uprop(mfl,k,i,j) = Uprop(mfl,k,i,j) -
     +        (tsfrac*dt/dx2)*
     +        (fjph(k,j)-fjph(k,j-1)) -
     +        tsfrac*dt*S(k)
      END DO
    END DO
  END DO
  END IF
  RETURN
  END
```

VORTICITY.f

```
  SUBROUTINE VORTICITY

  INCLUDE "commonblock"
C ---------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   04-06-2014: CREATED
C ---------------------------------------------------------------
C SUBROUTINE TO CALCULATE THE VORTICITY FIELD!!
c A basic calculation, but we should only bother doing it when we
c need to write some outputs...
C ---------------------------------------------------------------
```

```
      DOUBLE PRECISION dux1dx2,dux2dx1
C -----------------------------------------------------------------
C BASIC FIRST ORDER DERIVATIVE CALCULATION...
C -----------------------------------------------------------------
      DO i=2,nx1-1
        DO j=2,nx2-1
          dux1dx2 = (ux1(i,j+1)-ux1(i,j-1))/(2.0*dx2)
          dux2dx1 = (ux2(i+1,j)-ux2(i-1,j))/(2.0*dx1)
          IF (i.EQ.2)THEN
          dux2dx1 = (ux2(i+1,j)-ux2(i,j))/dx1
          END IF
          vort(i,j) = (1.0/x1(i))*(ux2(i,j) +
         +        x1(i)*dux2dx1 - dux1dx2)
        END DO
      END DO
      RETURN
      END
```

WENOZ.f

```
      SUBROUTINE WENOZ (fim2,fim1,fi,fip1,fip2,fimh,fiph)

      IMPLICIT NONE
C -----------------------------------------------------------------
C AUTHOR: J. R. C. KING
C ---------------------
C CHANGE RECORD:
C   ??-03-2013: CREATED
C -----------------------------------------------------------------
C SUBROUTINE TO CALCULATE FLUXES USING BORGES 5TH ORDER MODIFIED
C WENO SCHEME (as mentioned in Hu et al 2009)
C -----------------------------------------------------------------
C INPUT FUNCTION
      DOUBLE PRECISION  fim2,fim1,fi,fip1,fip2
C OUTPUT WENOZ APPROXIMATION
      DOUBLE PRECISION  fimh,fiph
C SMOOTHNESS INDICATORS
      DOUBLE PRECISION  beta0,beta1,beta2
      DOUBLE PRECISION  tau5,eps
      DOUBLE PRECISION  beta0z,beta1z,beta2z
C WEIGHTINGS
      DOUBLE PRECISION  d0,d1,d2
      DOUBLE PRECISION  alpha0z,alpha1z,alpha2z,sumalphaz
      DOUBLE PRECISION  w0z,w1z,w2z
C SMALL STENCIL INTERPOLATIONS
      DOUBLE PRECISION  fimh0,fimh1,fimh2,fiph0,fiph1,fiph2
C SET BASIC WEIGHTINGS ----------------------------------------
      d0=0.3
      d1=0.6
      d2=0.1
C CALCULATE INITIAL SMOOTHNESS INDICATORS ----------------------
      beta0 = (13.0/12.0)*(fim2-2.0*fim1+fi)**2.0 +
     +    0.25*(fim2-4.0*fim1+3.0*fi)**2.0
      beta1 = (13.0/12.0)*(fim1-2.0*fi+fip1)**2.0 +
     +    0.25*(fim1-fip1)**2.0
      beta2 = (13.0/12.0)*(fi-2.0*fip1+fip2)**2.0 +
     +    0.25*(3.0*fi-4.0*fip1+fip2)**2.0
      tau5 = abs(beta0-beta2)
C CALCULATE BETTER SMOOTHNESS INDICATORS -----------------------
      eps = 1e-40
      beta0z = (beta0+eps)/(beta0+tau5+eps)
      beta1z = (beta1+eps)/(beta1+tau5+eps)
      beta2z = (beta2+eps)/(beta2+tau5+eps)
C CALCULATE DIFFERENT WEIGHTINGS -------------------------------
      alpha0z = d0/beta0z
      alpha1z = d1/beta1z
```

```
47    alpha2z = d2/beta2z
      sumalphaz = alpha0z + alpha1z + alpha2z
49  C CALCULATE FINAL WEIGHTINGS -------------------------------------
      w0z = alpha0z/sumalphaz
51    w1z = alpha1z/sumalphaz
      w2z = alpha2z/sumalphaz
53  C CALCULATE THREE 3 POINT STENCIL ESTIMATES AT i+0.5 ------------
      fiph0 = fim2/3.0 - 7.0*fim1/6.0 + 11.0*fi/6.0
55    fiph1 = -1.0*fim1/6.0 + 5.0*fi/6.0 + fip1/3.0
      fiph2 = fi/3.0 + 5.0*fip1/6.0 - fip2/6.0
57  C AND A 5th ORDER ESTIMATE USING THE FINAL WEIGHTS --------------
      fiph = w0z*fiph0 + w1z*fiph1 + w2z*fiph2
59  C CALCULATE THREE 3 POINT STENCIL ESTIMATES AT i-0.5 ------------
      fimh0 = -1.0*fim2/6.0 + 5.0*fim1/6.0 + fi/3.0
61    fimh1 = fim1/3.0 + 5.0*fi/6.0 - fip1/6.0
      fimh2 = 11.0*fi/6.0 - 7.0*fip1/6.0 + fip2/3.0
63  C AND A 5th ORDER ESTIMATE USING THE FINAL WEIGHTS --------------
      fimh = w0z*fimh0 + w1z*fimh1 + w2z*fimh2
65    RETURN
      END
```

init.params

```
50              :nx1
50              :nx2
30000             :nt
200             :outfreq
1.0D0             :x1d
3.141D0             :x2d
0.8D0             :CFL
2.0D0             :coordsno
9.81D0              :grav
1             :viscflag
-1              :ghost?
7.7D3             :gun depth
0             :from_dump?
1             :boundflag:0wall,1NLAA
0             :distflag: wiggles?
```

AG.dat

```
102.0         :rho 1
8.85e6          :p 1
0.0         :ux1 1
0.0         :ux2 1
1.4         :gamma 1
0.0         :pc 1
1000.0          :rho 2
1.77e5          :p 2
0.0         :ux1 2
0.0         :ux2 2
7.0         :gamma 2
3.0e8         :pc 2
0.1         :Rint0
```

References

(1951). *Underwater Explosion Research: a compendium of British and American reports*. Office of Naval Research, United States Department of the Navy.

Abarbanel, S. and Gottlieb, D. (1997). A mathematical analysis of the PML method. *Journal of Computational Physics*, **134**, 357--363.

Adalsteinsson, D. and Sethian, J.A. (1995). A fast level set method for propagating interfaces. *Journal of Computational Physics*, **118**, 269--277.

Atassi, O.V. and Galan, J.M. (2008). Implementation of nonreflecting boundary conditions for the nonlinear Euler equations. *Journal of Computational Physics*, **227**, 1643--1662.

Balsara, D.S. and Shu, C.W. (2000). Monotonicity preserving weighted essentially non-oscillatory schemes with increasingly high order of accuracy. *Journal of Computational Physics*, **160**, 405--452.

Barker, D. and Landrø, M. (2012). Simple expression for the bubble-time period of two clustered air guns. *Geophysics*, **77**, A1--A3.

Barker, D. and Landrø, M. (2013). Estimation of bubble time period for air-gun clusters using potential isosurfaces. *Geophysics*, **78**, P1--P7.

Barker, D. and Landrø, M. (2014). An alternative method for modeling close-range interactions between air guns. *Geophysics*, **79**, P1--P7.

Barras, G., Souli, M., Aquelet, N. and Couty, N. (2012). Numerical simulation of underwater explosions using an ALE method. The pulsating bubble phenomena. *Ocean Engineering*, **41**, 53--66.

Bayliss, A. and Turkel, E. (1980). Radiation boundary conditions for wave-like equations. *Communications on Pure and Applied Mathematics*, **33**, 707--725.

Berenger, J.P. (1994). A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, **114**, 185--200.

Berenger, J.P. (1996). Three-dimensional perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, **127**, 363--379.

Blake, J.R. and Gibson, D.C. (1981). Growth and collapse of a vapour cavity near a free surface. *Journal of Fluid Mechanics*, **111**, 123--140.

Blake, J.R., Keen, G.S., Tong, R.P. and Wilson, M. (1999). Acoustic cavitation: the fluid dynamics of non-spherical bubbles. *Philosophical Transactions of the Royal Society of London A*, **357**, 251--267.

Borges, R., Carmona, M., Costa, B. and Don, W.S. (2008). An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. *Journal of Computational Physics*, **227**, 3191--3211.

Bornhorst, W.J. and Hatsopoulos, G.N. (1967). Bubble-growth calculation without neglect of interfacial discontinuities. *Journal of Applied Mechanics*, **34**, 847--853.

Bourne, N.K. and Field, J.E. (1992). Shock-induced collapse of single cavities in liquids. *Journal of Fluid Mechanics*, **244**, 225--240.

Brandsaeter, H., Farestveit, A. and Ursin, B. (1979). A new high-resolution or deep penetration airgun array. *Geophysics*, **44**, 865--879.

Chorin, A.J. (1976). Random choice solution of hyperbolic systems. *Journal of Computational Physics*, **22**, 517--533.

Cocchi, J.P., Saurel, R. and Loraud, J.C. (1996). Treatment of interface problems with Godunov-type schemes. *Shock Waves*, **5**, 347--357.

Colagrossi, A. and Landrini, M. (2003). Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics*, **191**, 448--475.

Cole, R.H. (1948). *Underwater Explosions*. Princeton University Press.

Colella, P. (1985). A direct Eulerian MUSCL scheme for gas dynamics. *SIAM Journal on Scientific and Statistical Computing*, **6**, 104--117.

Colella, P. and Woodward, P.R. (1984). The piecewise parabolic method (PPM) for gas-dynamical simulations. *Journal of Computational Physics*, **54**, 174--201.

Colonius, T. (2004). Modelling artificial boundary conditions for compressible flow. *Annual Review of Fluid Mechanics*, **36**, 315--345.

Cooke, C.H. and Chen, T.J. (1991). Continuous front tracking with subcell resolution. *Journal of Scientific Computing*, **6**, 269--282.

Courant, R., Friedrichs, K. and Lewy, H. (1928). ber die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, **100**, 32--74.

Courant, R., Friedrichs, K. and Lewy, H. (1967). On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, **11**, 215--234.

Cox, E., Pearson, A., Blake, J.R. and Otto, S.R. (2004). Comparison of methods for modelling the behaviour of bubbles produced by marine seismic airguns. *Geophysical Prospecting*, **52**, 451--477.

Dobratz, B.M. and Crawford, P.C. (1985). *LLNL explosives handbook: properties of chemical explosives and explosive simulants*. Lawrence Livermore National Laboratory.

Dragoset, W.H. (1984). A comprehensive method for evaluating the design of air guns and air gun arrays. *The Leading Edge*, **3**, 52--61.

Einfeldt, B. (1988). On Godunov-type methods for gas dynamics. *SIAM Journal of Numerical Analysis*, **25**, 294--318.

Engquist, B. and Majda, A. (1977). Absorbing boundary conditions for the numerical simulation of waves. *Mathematics of Computation*, **31**, 629--651.

Fedkiw, R.P. (2002). Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method. *Journal of Computational Physics*, **175**, 200--224.

Fedkiw, R.P. and Liu, X.D. (2001). The ghost fluid method for viscous flows. In M.M. Hafez and J.J. Chattot, eds., *Innovative methods for numerical solution of partial differential equations*, World Scientific.

Fedkiw, R.P., Aslam, T., Merriman, B. and Osher, S. (1999a). A non-oscillatory Eulerian approach to interfaces in multimaterial flows - the ghost fluid method. *Journal of Computational Physics*, **152**, 457--492.

Fedkiw, R.P., Marquina, A. and Merriman, B. (1999b). An isobaric fix for the overheating problem in multimaterial compressible flows. *Journal of Computational Physics*, **148**, 545--578.

Fine, R.A. and Millero, F.J. (1973). Compressibility of water as a function of temperature and pressure. *Journal of Chemical Physics*, **59**, 5529--5536.

Flores, J. and Holt, M. (1981). Glimm's method applied to underwater explosions. *Journal of Computational Physics*, **44**, 377--387.

Frisch, U. (1995). *Turbulence, The Lagacy of A. N. Kolmogorov*. Cambridge University Press.

Giles, B.F. and Johnston, R.C. (1973). System approach to air-gun array design. *Geophysical Prospecting*, **21**, 77--101.

Gilmore, F.R. (1952). Collapse of a spherical bubble. Tech. Rep. No. 26-4, Hydrodyn. Lab., Calif. Inst. Tech.

Givoli, D. (1991). Non-reflecting boundary conditions. *Journal of Computational Physics*, **94**, 1--29.

Glaister, P. (1988). An approximate linearised Riemann solver for the Euler equations for real gases. *Journal of Computational Physics*, **74**, 382--408.

Glimm, J. (1965). Solutions in the large for nonlinear hyperbolic systems of equations. *Communications on Pure and Applied Mathematics*, **18**, 697--715.

Godunov, S.K. (1959). A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, **47(89)**, 271--306.

Goncalvés, E. and Patella, R.F. (2011). Constraints on equation of state for cavitating flows with thermodynamic effects. *Applied Mathematics and Computation*, **217**, 5095--5102.

Grote, M.J. and Keller, J.B. (1995). Exact nonreflecting boundary conditions for the time dependent wave equation. *SIAM Journal on Applied Mathematics*, **55**, 280--297.

Hagstrom, T. (1999). Radiation boundary conditions for the numerical simulation of waves. *Acta Numerica*, **8**, 47--106.

Hagstrom, T. and Goodrich, J. (2003). Accurate radiation boundary conditions for the linearized Euler equations in Cartesian domains. *SIAM Journal on Scientific Computing*, **24**, 770--795.

Hagstrom, T. and Hariharan, S.I. (1988). Accurate boundary conditions for exterior problems in gas dynamics. *Mathematics of Computation*, **51**, 581--597.

Harten, A. (1983). High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, **49**, 357--393.

Harten, A. (1989). ENO schemes with subcell resolution. *Journal of Computational Physics*, **83**, 148--184.

Harten, A. and Lax, P.D. (1981). A random choice finite difference scheme for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, **18**, 289--315.

Harten, A., Lax, P.D. and van Leer, B. (1983). On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. *SIAM Review*, **25**, 35--61.

Hayder, M.E., Hu, F.Q. and Hussaini, M.Y. (1999). Towards perfectly absorbing boundary conditions for euler equations. *AIAA Journal*, **37**, 912--918.

Hedstrom, G.W. (1979). Nonreflecting boundary conditions for nonlinear hyperbolic systems. *Journal of Computational Physics*, **30**, 222--237.

Herring, C. (1941). *Theory of the pulsations of the gas bubble produced by an underwater explosion*. Office of Naval Research, Dept. of the Navy.

Higdon, R.L. (1986). Initial-boundary value problems for linear hyperbolic systems. *SIAM Review*, **28**, 177--217.

Holt, M. (1977). Underwater explosions. *Annual Review of Fluid Mechanics*, **9**, 187--214.

Hooton, M.C., Blake, J.R. and Soh, W.K. (1994). Behaviour of an underwater explosion bubble near a rigid boundary: theory and experiment. *Fluid Dynamics and Its Applications*, **23**, 421--428.

Hu, X.Y. and Khoo, B.C. (2004). An interface interaction method for compressible multifluids. *Journal of Computational Physics*, **198**, 35--64.

Hu, X.Y., Khoo, B.C., Adams, N.A. and Huang, F.L. (2006). A conservative interface method for compressible flows. *Journal of Computational Physics*, **219**, 553--578.

Hu, X.Y., Adams, N.A. and Iaccarino, G. (2009). On the HLLC Riemann solver for interface interaction in compressible multi-fluid flow. *Journal of Computational Physics*, **228**, 6572--6589.

Ivings, M.J., Causon, D.M. and Toro, E.F. (1998). On Riemann solvers for compressible liquids. *International Journal for Numerical Methods in Fluids*, **28**, 395--418.

Jiang, G.S. and Shu, C.W. (1996). Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, **126**, 202--228.

Johnson, D.T. (1994). Understanding air-gun bubble behaviour. *Geophysics*, **59**, 1729--1734.

Kane, J., Arnett, D., Remington, B.A., Glendinning, S.G., Bazan, G., Muller, E., Fryxell, B.A. and Teyssier, R. (2000). Two-dimensional versus three-dimensional supernova hydrodynamic instability growth. *The Astrophysical Journal*, **528**, 989--994.

Kang, M., Fedkiw, R.P. and Liu, X.D. (2000). A boundary condition capturing method for multiphase incompressible flow. *Journal of Scientific Computing*, **15**, 323--360.

Keller, J.B. and Kolodner, I.I. (1956). Damping of underwater explosion bubble oscillations. *Journal of Applied Physics*, **27**, 1152--1161.

Kifonidis, K., Plewa, T., Janka, H.T. and Muller, E. (2003). Non-spherical core collapse supernovae I. Neutrino-driven convection, Rayleigh-Taylor instabilities, and the formation and propagation of metal clumps. *Astronomy and Astrophysics*, **408**, 621--649.

Kifonidis, K., Plewa, T., Scheck, L., Janka, H.T. and Muller, E. (2006). Non-spherical core collapse supernovae II. The late-time evolution of globally anistropic neutrino-driven explosions and their implications for SN 1987 A. *Astronomy and Astrophysics*, **453**, 661--678.

King, J.R.C., Ziolkowski, A.M. and Ruffert, M. (2015). Artificial boundary conditions for simulations of oscillating bubbles using the non-linear acoustic approximation. *Journal of Computational Physics*, **284**, 273--290.

Kirkwood, J.G. and Bethe, H.A. (1942). Progress report on 'The pressure wave produced by an underwater explosion I'. Tech. Rep. No. 588, Office of Scientific Research and Development, US Navy.

Kucera, A. and Blake, J.R. (1990). Approximate methods for modelling cavitation bubbles near boundaries. *Bulletin of the Australian Mathematical Society*, **41**, 1--44.

Lamb, H. (1923). The early stages of submarine explosion. *Philosophical Magazine*, **45**, 257--265.

Landrø, M. (1992). Modelling of GI gun signatures. *Geophysical Prospecting*, **40**, 721--747.

Landrø, M. and Sollie, R. (1992). Source signature determination by inversion. *Geophysics*, **57**, 1633--1640.

Landrø, M., Amundsen, L. and Barker, D. (2011). High-frequency signals from air-gun arrays. *Geophysics*, **76**, Q19--Q27.

Langhammer, J. (1994). *Experimental studies of energy loss mechanisms in airgun bubble dynamics*. Ph.D. thesis, Norges Tekniske Hoegskole (NTNU), Trondheim.

Langhammer, J. and Landrø, M. (1993a). Experimental study of viscosity effects on air-gun signatures. *Geophysics*, **58**, 1801--1808.

Langhammer, J. and Landrø, M. (1993b). Temperature effects on airgun signatures. *Geophysical Prospecting*, **41**, 737--750.

Langhammer, J. and Landrø, M. (1996). High-speed photography of the bubble generated by an airgun. *Geophysical Prospecting*, **44**, 153--172.

Langhammer, J., Graciet, S., Vik, I., Espeland, M. and Lokberg, O.J. (1995). Holographic studies of the bubble generated by a seismic airgun. *Journal of the Acoustical Society of America*, **97**, 362--369.

Laws, R.M., Hatton, L. and Haartsen, M. (1990). Computer modelling of clustered airguns. *First Break*, **8**, 331--338.

Lax, P. and Wendroff, B. (1960). Systems of conservation laws. *Communications on Pure and Applied Mathematics*, **13**, 217--237.

Lax, P.D. (1954). Weak solutions of nonlinear hyperbolic equations and their numerical computation. *Communications on Pure and Applied Mathematics*, **7**, 159--193.

Lezzi, A. and Prosperetti, A. (1987). Bubble dynamics in a compressible liquid. Part 2. Second-order theory. *Journal of Fluid Mechanics*, **185**, 289--321.

Li, G.F., Cao, M.Q., Chen, H.L. and Ni, C.Z. (2010). Modeling air gun signatures in marine seismic exploration considering multiple physical factors. *Applied Geophysics*, **7**, 158--165.

Liou, M.S. (1996). A sequel to AUSM: AUSM+. *Journal of Computational Physics*, **129**, 364--382.

Liou, M.S. and Steffen, C.J. (1993). A new flux splitting scheme. *Journal of Computational Physics*, **107**, 23--39.

Liu, T.G., Khoo, B.C. and Yeo, K.S. (2001a). The simulation of compressible multimedium flow. I. A new methodology with test applications to 1D gas-gas and gas-water cases. *Computers and Fluids*, **30**, 291--314.

Liu, T.G., Khoo, B.C. and Yeo, K.S. (2001b). The simulation of compressible multi-medium flow II. Applications to 2D underwater shock refraction. *Computers and Fluids*, **30**, 315--337.

Liu, T.G., Khoo, B.C. and Yeo, K.S. (2003). Ghost fluid method for strong shock impacting on material interface. *Journal of Computational Physics*, **190**, 651--681.

Liu, T.G., Khoo, B.C. and Wang, C.W. (2005). The ghost fluid method for compressible gas-water simulation. *Journal of Computational Physics*, **204**, 193--221.

Liu, X.D., Osher, S. and Chan, T. (1994). Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, **115**, 200--212.

Luo, H., Baum, J.D. and Löhner, R. (2004). On the computation of multi-material flows using ALE formulation. *Journal of Computational Physics*, **194**, 304--328.

Martynov, S.B., Mason, D.J. and Heikal, M.R. (2006). Numerical simulation of cavitation flows based on their hydrodynamic similarity. *International Journal of Engine Research*, **7**, 283--296.

Mattssonn, A., Parkes, G. and Hedgeland, D. (2012). Svein vaage broadband airgun study. In A.N. Popper and A. Hawkins, eds., *The Effects of Noise on Aquatic Life*, vol. 730 of *Advances in Experimental Medicine and Biology*, 469--471, Springer New York.

Miller, S.T., Jasak, H., Boger, D.A., Paterson, E.G. and Nedungadi, A. (2013). A pressure-based, compressible, two-phase flow finite volume method for underwater explosions. *Computers and Fluids*, **87**, 132--143.

Muller, S., Bachmann, M., Krninger, D., Kurz, T. and Helluy, P. (2009). Comparison and validation of compressible flow simulations of laser-induced cavitation bubbles. *Computers and Fluids*, **38**, 1850--1862.

Neumann, A.W., David, R. and Zuo, Y., eds. (2010). *Applied Surface Thermodynamics*. CRC Press, 2nd edn.

Osher, S. and Sethian, J.A. (1988). Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, **79**, 12--49.

Park, S.H. and Kwon, J.H. (2003). On the dissipation mechanisms of Godunov-type schemes. *Journal of Computational Physics*, **188**, 524--542.

Parkes, G.E., Ziolkowski, A.M., Hatton, L. and Haugland, T. (1984). The signature of an air gun array: Computation from near-field measurements including interactions - Practical considerations. *Geophysics*, **48**, 105--111.

Pishevar, A.R. and Amirifar, R. (2010). An adaptive ALE method for underwater explosion simulations including cavitation. *Shock Waves*, **20**, 425--439.

Plesset, M.S. (1949). The dynamics of caviation bubbles. *Journal of Applied Mechanics*, **16**, 277--282.

Plesset, M.S. and Zwick, S.A. (1952). A nonsteady heat diffusion problem with spherical symmetry. *Journal of Applied Physics*, **23**, 95--98.

Poole, G. and Davison, C. (2013). Shot-to-shot directional designature using near-field hydrophone data. In *SEG Technical Program Expanded Abstracts 2013*, 4236--4240.

Prosperetti, A. and Lezzi, A. (1986). Bubble dynamics in a compressible liquid. Part 1. First-order theory. *Journal of Fluid Mechanics*, **168**, 457--478.

Prosperetti, A. and Plesset, M.S. (1978). Vapour-bubble growth in a superheated liquid. *Journal of Fluid Mechanics*, **85**, 349--368.

Rayleigh, J.W.S. (1917). On the pressure developed in a liquid during the collapse of a spherical cavity. *Philosophical Magazine*, **34**, 94--99.

Robinson, J.C. (2004). *An introduction to ordinary differential equations*. Cambridge University Press.

Roe, P.L. (1986). Characteristic-based schemes for the Euler equations. *Annual Review of Fluid Mechanics*, **18**, 337--365.

Roe, P.L. (1997). Approximate Riemann solvers, parameter vectors and difference schemes. *Journal of Computational Physics*, **135**, 250--258.

Russo, G. and Smereka, P. (2000). A remark on computing distance functions. *Journal of Computational Physics*, **163**, 51--67.

Shaw, S.J. and Spelt, P.D.M. (2010). Shock emission from collapsing gas bubbles. *Journal of Fluid Mechanics*, **646**, 363--373.

Shu, C.W. and Osher, S. (1988). Efficient implementation of essentially non-oscillatory shock capturing schemes. *Journal of Computational Physics*, **77**, 439--741.

Shu, C.W. and Osher, S. (1989). Efficient implementation of essentially non-oscillatory shock capturing schemes, II. *Journal of Computational Physics*, **83**, 32--78.

Smith, R.W. (1999). AUSM(ALE): A geometrically conservative arbitrary Lagrangian-Eulerian flux splitting scheme. *Journal of Computational Physics*, **150**, 268--286.

Sod, G.A. (1978). A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, **27**, 1--31.

Spelt, P.D.M. (2005). A level-set approach for simulations of flows with multiple moving contact lines with hysteresis. *Journal of Computational Physics*, **207**, 389--404.

Strang, G. (1968). On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, **5**, 506--517.

Sutherland, W. (1893). The viscosity of gases and molecular force. *Philosophical Magazine Series 5*, **36**, 507--531.

Taylor, G.I. (1942). *Vertical motion of a spherical bubble and the pressure surrounding it*. Office of Naval Research, Dept. of the Navy.

Taylor, G.I. (1950). The instability of liquid surfaces when accelerated in a direction perpendicular to their planes. I. *Proceedings of the Royal Society of London A*, **201**, 192--196.

Taylor, G.I. and Davies, R.M. (1943). *The motion and shape of the hollow produced by an explosion in a liquid*. Office of Naval Research, Dept. of the Navy.

Terashima, H. and Tryggvason, G. (2009). A front-tracking/ghost fluid method for fluid interfaces in compressible flows. *Journal of Computational Physics*, **228**, 4012--4037.

Thompson, K.W. (1987). Time dependent boundary conditions for hyperbolic systems. *Journal of Computational Physics*, **68**, 1--24.

Thompson, K.W. (1990). Time dependent boundary conditions for hyperbolic systems, II. *Journal of Computational Physics*, **89**, 439--461.

Tomita, Y. and Shima, A. (1986). Mechanisms of impulsive pressure generation and damage pit formation by bubble collapse. *Journal of Fluid Mechanics*, **169**, 535--564.

Toro, E.F., Spruce, M. and Speares, M. (1994). Restoration of the contact surface in the HLL-Riemann solver. *Shock Waves*, **4**, 25--34.

Tritton, D.J. (1988). *Physical Fluid Dynamics*. Clarendon Press, 2nd edn.

Tsynkov, S.V. (1998). Numerical solution of problems on unbounded domains. A review. *Applied Numerical Mathematics*, **27**, 465--532.

Turkel, E. and Yefet, A. (1998). Absorbing PML boundary layers for wave-like equations. *Applied Numerical Mathematics*, **27**, 533--557.

van Leer, B. (1974). Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme. *Journal of Computational Physics*, **14**, 361--370.

van Leer, B. (1979). Towards the ultimate conservative difference scheme V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, **135**, 229--248.

van Leer, B. (1982). Flux-vector splitting for the euler equations. *Lecture Notes in Physics*, **170**, 507--512.

Wagner, W. and Pruss, A. (1993). International equations for the saturation properties of ordinary water substance. Revised according to the international temperature scale of 1990. Addendum to J. Phys. Chem. Ref. Data 16, 893 (1987). *Journal of Physical and Chemical Reference Data*, **22**, 783--787.

Wang, W., Liu, T.G. and Khoo, B.C. (2006). A real ghost fluid method for the simulation of multimedium compressible flow. *SIAM Journal of Scientific Computing*, **28**, 278--302.

Wardlaw, A.B. and Mair, H.U. (1998). Spherical solutions of an underwater explosion bubble. *Shock and Vibration*, **5**, 89--102.

Wilkerson, S. (1992). A boundary integral approach for three-dimensional underwater explosion bubble dynamics. Tech. Rep. BRL-TR-3365, Ft. Belvoir: Defense Technical Information Center.

Xu, K. and Li, Z. (2001). Dissipative mechanisms in Godunov-type schemes. *International Journal for Numerical Methods in Fluids*, **37**, 1--22.

Yuan, W., Sauer, J. and Schnerr, G.H. (2001). Modeling and computation of unstedy cavitation flows in injection nozzles. *Mecanique et Industries*, **2**, 383--394.

Zhang, A., Yang, W.S., Huang, C. and Ming, F. (2013a). Numerical simulation of column charge underwater explosion based on SPH and BEM combination. *Computers and Fluids*, **71**, 169--178.

Zhang, A.M., Wang, S.P. and Wu, G.X. (2013b). Simulation of bubble motion in a compressible liquid based on the three dimensional wave equation. *Engineering Analysis with Boundary Elements*, **37**, 1179--1188.

Ziolkowski, A. (1970). A method for calculating the output pressure waveform from an air gun. *Geophysical Journal of the Royal Astronomical Society*, **21**, 137--161.

Ziolkowski, A. (1998). Measurement of air-gun bubble oscillations. *Geophysics*, **63**, 2009--2024.

Ziolkowski, A.M. (1986). 3D measurement and evaluation of marine seismic sources. Tech. Rep. T.02.06/86, Tulip Geophysical BV.

Ziolkowski, A.M. and Johnston, R.G.K. (1997). Marine seismic sources: QC of wavefield computation from near-field pressure measurements. *Geophysical Prospecting*, **45**, 611--639.

Ziolkowski, A.M. and Metselaar, G. (1984). The pressure wavefield of an air gun array. In *Expanded Abstracts 54th SEG Meeting, Atlanta*, 274--276.

Ziolkowski, A.M., Parkes, G.E., Hatton, L. and Haugland, T. (1982). The signature of an air gun array; computation from near-field measurements including interactions. *Geophysics*, **47**, 1413--1421.

Journal of Computational Physics 284 (2015) 273–290

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Boundary conditions for simulations of oscillating bubbles using the non-linear acoustic approximation

J.R.C. King [a,*], A.M. Ziolkowski [a], M. Ruffert [b]

[a] University of Edinburgh, School of GeoSciences, Grant Institute, The King's Buildings, James Hutton Road, Edinburgh, EH9 3FE, UK
[b] University of Edinburgh, School of Mathematics & Maxwell Institute, JCMB, The King's Buildings, Peter Guthrie Tait Road, Edinburgh, EH9 3FD, UK

ARTICLE INFO

Article history:
Received 10 February 2014
Received in revised form 27 November 2014
Accepted 21 December 2014
Available online 29 December 2014

Keywords:
Boundary condition
Non-linear acoustic approximation
Underwater explosion
Oscillating bubble
Ghost fluid method

ABSTRACT

We have developed a new boundary condition for finite volume simulations of oscillating bubbles. Our method uses an approximation to the motion outside the domain, based on the solution at the domain boundary. We then use this approximation to apply boundary conditions by defining incoming characteristic waves at the domain boundary. Our boundary condition is applicable in regions where the motion is close to spherically symmetric. We have tested our method on a range of one- and two-dimensional test cases. Results show good agreement with previous studies. The method allows simulations of oscillating bubbles for long run times (5×10^5 time steps with a CFL number of 0.8) on highly truncated domains, in which the boundary condition may be applied within 0.1% of the maximum bubble radius. Conservation errors due to the boundary conditions are found to be of the order of 0.1% after 10^5 time steps. The method significantly reduces the computational cost of fixed grid finite volume simulations of oscillating bubbles. Two-dimensional results demonstrate that highly asymmetric bubble features, such as surface instabilities and the formation of jets, may be captured on a small domain using this boundary condition.

1. Introduction

Many problems in fluid dynamics are posed in unbounded domains. Various methods are employed to enable these problems to be solved numerically, a common one of which is to restrict the computation to a finite region and impose artificial boundary conditions on the truncated domain. The aim of the artificial boundary conditions is to mimic the unbounded domain and prevent spurious reflections from the domain boundary. Artificial boundary conditions of this type are often referred to as 'non-reflecting' and 'absorbing'. Inaccurate absorbing boundary conditions can lead to spurious disturbances at the domain boundaries, which propagate back through the domain, contaminating results.

There has been much work on non-reflecting boundary conditions. Reviews have been provided by Givoli [1], Hagstrom [2] and Tsynkov [3]. Many methods have been derived for wave propagation problems, such as the perfectly matched layer method [4,5] which are only strictly applicable to linear hyperbolic systems. However similar methods for the Euler equations have been developed [6]. A good review of work on artificial boundary conditions for compressible flow is given by Colonius [7].

* Corresponding author. Tel.: +44 (0) 131 650 5916.
E-mail address: j.r.c.king@ed.ac.uk (J.R.C. King).

http://dx.doi.org/10.1016/j.jcp.2014.12.037

Hedstrom [8] decomposed the Euler equations into characteristic wave equations and presented a non-reflecting boundary condition using these equations. Thompson [9] developed a useful formalism for applying characteristic boundary conditions, and described a method of applying non-reflecting characteristic boundary conditions [10]. The characteristic boundary condition formalism of [9] is widely used.

The behaviour of oscillating air bubbles in water is of interest in a variety of fields, including cavitation, underwater explosions, and shock wave lithotripsy. In 1917, Rayleigh [11] developed an equation of motion for a spherical cavity in an infinite incompressible fluid. The analysis in [11] forms the basis for much work on oscillating bubbles. Lamb [12] derived an exact wave equation for the motion of a spherical cavity in a compressible fluid, and obtained an analytical solution for the special case of the ratio of specific heats, γ, being equal to 4/3. In general there is no analytical solution to this equation. Extensions to [11] accounting for the compressibility of water were developed by several authors [13,14]. Gilmore [15] developed the work of [11–13] to obtain an algorithm to calculate the propagating wavefield outside the bubble. This scheme requires further approximations, and is less accurate than the equations of motion on which it is based. Although on a much smaller scale, cavitation bubbles are a physically similar phenomenon, and similar approximations have been developed to describe cavitation bubbles [16,17], leading to the widely used Rayleigh–Plesset equation. Further analysis of this form of approximation was carried out in [18,19].

Marine seismic exploration can be thought of as a powerful form of echo sounding, capable of penetrating the sea floor, to enable a three-dimensional image of the sub-sea to be created. It is a process used in the petroleum industry in the search for geological features which have the potential to contain trapped hydrocarbons. Initially dynamite was used as the source in marine seismic exploration. However environmental concerns led to the development of alternative sources. Currently, seismic air guns are the most commonly used source. In use they are towed behind a ship, usually between 5 and 20 metres beneath the sea surface, and when 'fired' release a quantity of air at high pressure (136 atm), that forms a bubble which oscillates, producing a wavefield which propagates through the sea and into the subsurface. A seismic air gun is analogous to a weak underwater explosion.

Air gun bubbles were first modelled in 1970 by Ziolkowski [20], using a simplified two-equation ordinary differential equation model of a seismic air gun based on the work of [15]. This method is still the basis of the modelling currently used by industry. The non-linear acoustic approximation (NLAA) was developed by Ziolkowski [21] as an improvement to [20], and is equivalent to the approximations of [13,14]. The NLAA approximates the wavefield produced by an oscillating bubble – subject to certain assumptions – and allows the calculation of pressure and velocity at any point provided the pressure and velocity are known at a single location. Boundary integral methods which allowed the simulation of non-spherical bubbles have been developed for cavitation modelling [22–24]. Cox et al. [25] provide a good review of air gun modelling, and apply earlier boundary integral methods to seismic air guns.

The first finite volume simulations of underwater explosions appear in [26], although a lack of adequate boundary conditions mean that only very early stages of the explosion were calculated. A review of early work on underwater explosions is provided in [27]. The one-dimensional spherically symmetric underwater explosion has since become a commonly used test case (although no analytical solution exists) for multimedium Euler solvers, and has been simulated using a variety of numerical schemes [28–33]. In a typical underwater explosion problem the outgoing pressure wavefield propagates to approximately 100 times the maximum bubble radius during a single bubble oscillation. Previous simulations have relied either on the use of a very large domain (for example [33]) which is computationally expensive, or on arbitrary Lagrangian–Eulerian methods [29–31] in which the problem is solved on a mesh which expands to contain the outgoing wavefield.

Our solution is to take the non-linear acoustic approximation and use it to develop artificial boundary conditions for a finite volume simulation of an oscillating bubble on a truncated domain. We base our approximation on the conditions at the domain boundary. This approximation is then used to describe any incoming characteristic waves. These characteristic waves are then applied through the characteristic boundary condition formalism of [9]. This method allows finite volume simulations of oscillating bubbles to be carried out for long run times on comparatively small domains, reducing computational costs. We present our theory in two dimensions, and provide one- and two-dimensional results, although the theory could be applied to three-dimensional simulations.

The layout of the paper is as follows. In Section 2 we show the derivation of the NLAA. In Section 3 we present a brief summary of the characteristic boundary condition formalism and use the NLAA to derive artificial boundary conditions. In Section 4 we describe the numerical scheme in which we implement our boundary conditions. In Section 5 we present the results of some one-dimensional test cases using our new method, and discuss the performance of the method. In Section 6 we present results from two-dimensional simulations. Section 7 is a summary of conclusions.

2. The non-linear acoustic approximation

Ziolkowski [21] developed the non-linear acoustic approximation for the motion of a spherical bubble in water for use in modelling seismic air guns. The approximation is based on the assumption that the acoustic wavefield produced by the bubble is dominated by wavelengths many times the bubble diameter, which allows the bubble to be considered a monopole source. The velocity is described by a velocity potential which is assumed to obey the linear acoustic wave equation, leading to an analytical solution for the velocity potential. This solution is then passed back into the Euler equations to obtain solutions for the pressure and velocity. The following is the derivation from [21]. We include this derivation as we will refer

to several of the equations frequently in the following sections of this paper, and wish to avoid cumbersome citations to facilitate flow of exposition.

Starting from Lamb [12], the bubble is assumed to be spherical, and all motion is in the radial direction and subject to spherical symmetry. The local specific enthalpy in the water, h, is defined as

$$h = \int_{p_\infty}^{p} \frac{dp}{\rho} = \int_{\rho_\infty}^{\rho} c^2 \frac{d\rho}{\rho}, \tag{1}$$

where p is the pressure, ρ is the density, and c is the speed of sound, defined by $c^2 = \frac{dp}{d\rho}\big|_{isentropic}$. p_∞ and ρ_∞ are the pressure and density in the undisturbed water. For the pressure fluctuations considered, it is acceptable to assume that $\rho = \rho_\infty$ and $h = (p - p_\infty)/\rho_\infty$. The speed of sound in the water is assumed to be constant. It should be noted that this combination of assumptions – incompressible flow and finite speed of sound – is a contradictory set of assumptions, but acceptable because of the low Mach number flows involved. Viscosity is neglected [20]. The flow is assumed to be irrotational and the velocity $\mathbf{u} = u\mathbf{e_r}$ obeys a velocity potential such that $\mathbf{u} = -\nabla\phi$. Hence $u = -\frac{\partial\phi}{\partial r}$. The equation of motion is written

$$\frac{D\mathbf{u}}{Dt} + \nabla h = 0, \tag{2}$$

where $\frac{D(\cdot)}{Dt}$ is the material derivative, defined by $\frac{D(\cdot)}{Dt} = \frac{\partial(\cdot)}{\partial t} + u\nabla(\cdot)$. Eq. (2) is integrated to give

$$h = \frac{\partial\phi}{\partial t} - \frac{u^2}{2}. \tag{3}$$

An equation for the conservation of mass is written

$$\frac{1}{\rho}\frac{D\rho}{Dt} - \nabla^2\phi = 0. \tag{4}$$

Eqs. (1) and (4) are combined to obtain

$$\frac{Dh}{Dt} = \frac{c^2}{\rho}\frac{D\rho}{Dt} = c^2\nabla^2\phi. \tag{5}$$

With the imposition of spherical symmetry, from Eqs. (3) and (5) the exact wave equation, first derived in [12], is obtained

$$\frac{\partial^2\phi}{\partial r^2}\left(1 - \frac{u^2}{c^2}\right) + \frac{2}{r}\frac{\partial\phi}{\partial r}\left(1 + \frac{r}{c^2}\frac{\partial^2\phi}{\partial t\partial r}\right) - \frac{1}{c^2}\frac{\partial^2\phi}{\partial t^2} = 0. \tag{6}$$

Eq. (6) has no known analytical solution. If the advective terms – $u\frac{\partial(\cdot)}{\partial r}$ – in Eqs. (2), (4) and (5) are neglected the linear acoustic wave equation in ϕ is obtained

$$\frac{\partial^2\phi}{\partial r^2} + \frac{2}{r}\frac{\partial\phi}{\partial r} - \frac{1}{c^2}\frac{\partial^2\phi}{\partial t^2} = 0. \tag{7}$$

Lamb [12] estimated that for flows with Mach number less than approximately 0.1, the errors caused by this approximation would be less than 1%. The wavelengths of the pressure field produced by the bubble are large compared with the bubble radius, hence the bubble can be considered a point source. There are no other sources. Under these conditions, Eq. (7) has the well known solution

$$\phi(r,t) = \frac{1}{r}f\left(t - \frac{r}{c}\right). \tag{8}$$

Differentiation of Eq. (8) yields

$$u(r,t) = -\frac{\partial\phi}{\partial r} = \frac{1}{r^2}f + \frac{1}{rc}f' \tag{9}$$

where the argument of f, $(t - r/c)$, has been dropped for ease of writing, and a prime denotes differentiation with respect to the argument. Further differentiation gives

$$\frac{\partial u}{\partial r} = -\frac{\partial^2\phi}{\partial r^2} = \frac{-2}{r^3}f - \frac{2}{r^2c}f' - \frac{1}{rc^2}f'' \tag{10}$$

and

$$\frac{\partial u}{\partial t} = -\frac{\partial^2\phi}{\partial r\partial t} = \frac{1}{r^2}f' + \frac{1}{rc}f''. \tag{11}$$

These results are passed into Eqs. (2) and (4) giving

$$\frac{1}{r^2}f' + \frac{1}{rc}f'' + \frac{1}{\rho}\frac{\partial p}{\partial r} + u\left(\frac{-2}{r^3}f - \frac{2}{r^2c}f' - \frac{1}{rc^2}f''\right) = 0. \tag{12}$$

Eqs. (3), (8), (9) and (11) are used to provide expressions for f, f' and f''

$$f = r^2\left(u - \frac{h}{c} - \frac{u^2}{2c}\right), \tag{13}$$

$$f' = r\frac{\partial \phi}{\partial t} = r\left(h + \frac{u^2}{2}\right), \tag{14}$$

$$f'' = rc\frac{\partial u}{\partial t} - ch - \frac{cu^2}{2}. \tag{15}$$

Ziolkowski [21] argues that the quantity $r(h + u^2/2)$ propagates outwards at speed c with uniform amplitude, and determines a Lagrangian form of this result, obtaining

$$R\ddot{R}\left(1 - \frac{2\dot{R}}{c}\right) + \frac{3\dot{R}^2}{2}\left(1 - \frac{4\dot{R}}{3c}\right) = H + \frac{R\dot{H}}{c}\left(1 - \frac{\dot{R}}{c}\right), \tag{16}$$

where R is the bubble radius, H the enthalpy of the water at the bubble wall, c the speed of sound of water. A dot represents differentiation with respect to time. This result is solved numerically along with an equation such as $PR^{3n} = constant$, where n is a constant and $1 \leq n \leq 1.4$ [20], to simulate the evolution of the bubble through time. P and R are also used with the above results to calculate an approximation of the pressure and velocity at any point in the water.

3. Definition of boundary conditions using the non-linear acoustic approximation

3.1. Characteristic boundary condition formalism

Consider a finite spherical domain Ω, of radius R_D, bounded by Γ. Ω is centred on the origin of a polar coordinate system, subject to polar axisymmetry, in which r is the radial distance and θ the polar angle. Thompson [9] presents a formalism for the treatment of boundary conditions in finite difference simulations for hyperbolic systems of conservation laws. This method decomposes of the system of equations into a set of uncoupled wave equations for non-linear characteristics. This set of equations is then solved on domain boundaries, with any incoming characteristic waves being specified according to the boundary condition desired. Starting from the Euler equations for primitive variables density ρ, pressure p, radial velocity u and polar velocity v, in polar coordinates, the uncoupled wave equations for non-linear characteristics on Γ may be written as

$$\frac{\partial \rho}{\partial t} + \frac{1}{c^2}\left\{\mathscr{L}_2 + \frac{1}{2}[\mathscr{L}_4 + \mathscr{L}_1]\right\} + \frac{2\rho u}{r} + \frac{v}{r}\frac{\partial \rho}{\partial \theta} + \frac{\rho}{r}\frac{\partial v}{\partial \theta} + \frac{\rho v}{r\tan\theta} = 0, \tag{17}$$

$$\frac{\partial p}{\partial t} + \frac{1}{2}\{\mathscr{L}_4 + \mathscr{L}_1\} + \frac{2\rho c^2 u}{r} + \frac{v}{r}\frac{\partial p}{\partial \theta} + \frac{\rho c^2}{r}\frac{\partial v}{\partial \theta} + \frac{\rho v c^2}{r\tan\theta} = 0, \tag{18}$$

$$\frac{\partial u}{\partial t} + \frac{1}{2\rho c}\{\mathscr{L}_4 - \mathscr{L}_1\} + \frac{v}{r}\frac{\partial u}{\partial \theta} + \frac{v^2}{r} + g\cos\theta = 0, \tag{19}$$

$$\frac{\partial v}{\partial t} + \mathscr{L}_3 + \frac{v}{r}\frac{\partial v}{\partial \theta} + \frac{1}{r\rho}\frac{\partial p}{\partial \theta} + \frac{uv}{r} - g\sin\theta = 0, \tag{20}$$

where each of the four $\mathscr{L}_i$ describes a characteristic wave mode, each with propagation speed λ_i. The terms proportional to $2u/r$ and $1/r\tan\theta$ are source terms due to the polar coordinates. c is the local speed of sound. The λ_i are defined by

$$\lambda_1 = u - c, \qquad \lambda_2 = \lambda_3 = u, \qquad \lambda_4 = u + c. \tag{21}$$

The $\mathscr{L}_i$ are defined as

$$\mathscr{L}_1 = \lambda_1\left\{\frac{\partial p}{\partial r} - \rho c\frac{\partial u}{\partial r}\right\}, \tag{22}$$

$$\mathscr{L}_2 = \lambda_2\left\{c^2\frac{\partial \rho}{\partial r} - \frac{\partial p}{\partial r}\right\}, \tag{23}$$

$$\mathscr{L}_3 = \lambda_3\left\{\frac{\partial v}{\partial r}\right\}, \tag{24}$$

$$\mathscr{L}_4 = \lambda_4\left\{\frac{\partial p}{\partial r} + \rho c\frac{\partial u}{\partial r}\right\}, \tag{25}$$

Finite difference simulations of the Euler equations involve the following scheme: (1) calculation of spatial derivatives based on the solution at the current time step; (2) using these spatial derivatives in some form of the Euler equations to determine time derivatives; then (3) integrating the time derivatives to obtain the solution at the next time step. On boundary nodes equations (17) to (20) are solved. At any point in space, each characteristic wave mode $\mathscr{L}_i$ is described entirely by information downstream of that point determined by the corresponding characteristic wave speed λ_i.

If the characteristic wave $\mathscr{L}_i$ is propagating into Ω on Γ ($\lambda_i < 0$ at $r = R_D$), the information describing that wave mode is contained entirely outside the domain, and hence $\mathscr{L}_i$ must be specified by some artificial boundary condition (for example, $\mathscr{L}_i = 0$). If the characteristic wave is propagating out of the domain then it is entirely defined by information contained within the domain, in which case Eqs. (22), (23), (24) or (25) may be used to define $\mathscr{L}_i$, based upon the solution within the domain. For example, a zero-velocity boundary condition (a reflective boundary) on Γ is applied by computing $\mathscr{L}_4$ from its definition in Eq. (25), prescribing $\mathscr{L}_1 = \mathscr{L}_4$ and $\mathscr{L}_2 = \mathscr{L}_3 = 0$, and then solving equations (17) to (20) at that point.

Thompson [9] describes a 'non-reflecting' boundary condition as one in which all characteristic waves incoming to the domain are suppressed. To apply this boundary condition on Γ with subsonic flow, we compute $\mathscr{L}_4$ from its definition in Eq. (25) and prescribe $\mathscr{L}_1 = 0$. $\mathscr{L}_2 = \mathscr{L}_3 = 0$ if $u(b,t) \geq 0$, otherwise $\mathscr{L}_2$ and $\mathscr{L}_3$ are defined from Eqs. (23) and (24). We then solve equations (17) to (20) at that point. Thompson [10] admits that there are many situations in which the correct solution does contain both outgoing and incoming characteristic waves, and demonstrates some of the limitations of this boundary condition.

3.2. Boundary conditions using the non-linear acoustic approximation

The NLAA yields a good approximation to the motion of an air gun bubble or underwater explosion provided it is not used in such close proximity to the bubble that the assumptions on which is founded are invalid. If an oscillating bubble is simulated on a finite domain of sufficient radius, then the approximate motion of the water outside the domain may be calculated using the NLAA based on the solution on the domain boundary. Furthermore, this approximate solution may then be used to provide boundary conditions for the finite volume simulation of the bubble.

The NLAA is only valid for problems with spherical symmetry, in regions where the density variation is small, and velocities are small compared with sound speeds. Consider again the domain Ω, defined by $0 \leq r \leq R_D$, bounded by Γ. Within Ω there may be an air gun bubble, an underwater explosion, or some other source, but R_D is large enough that on Γ the NLAA is valid. Within Ω there is no limit to flow speeds or directions. On R_D, $\lambda_1 < 0$ and $\lambda_4 > 0$. As such, $\mathscr{L}_1$ must be specified on the boundary from information based on the approximation to the exterior flow. The velocities in the water will sometimes be directed inwards and sometimes outwards. If $u(R_D,t) > 0$ then $\mathscr{L}_2$ and $\mathscr{L}_3$ can be calculated from Eqs. (23) and (24). If $u(R_D,t) < 0$ then $\mathscr{L}_2$ and $\mathscr{L}_3$ must be specified from information based on the external flow. $\mathscr{L}_4$ will always be set by Eq. (25).

3.2.1. Prescription of $\mathscr{L}_1$

The results of Eqs. (10) and (12) are passed back into the definition of $\mathscr{L}_1$ to obtain

$$\mathscr{L}_1 = \frac{\rho(u-c)}{r}\left\{-\frac{1}{r}f' - \frac{1}{c}f'' + (u+c)\left(\frac{2}{r^2}f + \frac{2}{rc}f' + \frac{1}{c^2}f''\right)\right\}. \tag{26}$$

Using the results of Eqs. (13), (14) and (15) Eq. (26) becomes

$$\mathscr{L}_1 = \frac{\rho(u-c)}{r}\left\{\frac{u^2}{2}\left(3 - \frac{u}{c}\right) + 2uc - \frac{p-p_\infty}{\rho_\infty}\left(1 + \frac{u}{c}\right) + \frac{u}{c}r\frac{\partial u}{\partial t}\right\}. \tag{27}$$

We write

$$\mathscr{L}_1 = \frac{\rho(u-c)u}{c}\frac{\partial u}{\partial t} + \alpha_{\mathscr{L}_1}, \tag{28}$$

where

$$\alpha_{\mathscr{L}_1} = \frac{\rho(u-c)}{r}\left\{\frac{u^2}{2}\left(3 - \frac{u}{c}\right) + 2uc - \frac{p-p_\infty}{\rho_\infty}\left(1 + \frac{u}{c}\right)\right\}. \tag{29}$$

Eq. (19) may now be expressed as

$$\frac{\partial u}{\partial t} + \frac{1}{2\rho c}\left\{\mathcal{L}_4 - \alpha_{\mathscr{L}_1} - \frac{\rho(u-c)u}{c}\frac{\partial u}{\partial t}\right\} + \frac{v}{r}\frac{\partial v}{\partial \theta} + \frac{v^2}{r} + g\cos\theta = 0. \tag{30}$$

Eq. (30) can be re-arranged to form

$$\frac{\partial u}{\partial t} + \frac{\frac{1}{2\rho c}\{\mathscr{L}_4 - \alpha_{\mathscr{L}_1}\} + \frac{v}{r}\frac{\partial v}{\partial \theta} + \frac{v^2}{r} + g\cos\theta}{1 - \frac{(u-c)u}{2c^2}} = 0 \tag{31}$$

Eq. (31) is solved to find $\frac{\partial u}{\partial t}$. Eq. (28) is then used to calculate $\mathscr{L}_1$, which is passed to Eqs. (17) and (18), and used to calculate $\frac{\partial \rho}{\partial t}$ and $\frac{\partial p}{\partial t}$.

3.2.2. Prescription of $\mathscr{L}_2$

When $u > 0$ on Γ, $\mathscr{L}_2$ may be determined from the definition in Eq. (23). When $u < 0$ on Γ, $\mathscr{L}_2$ must be determined based on the solution of the NLAA. The NLAA is based on the contradictory combination of assumptions of constant finite sound speed in an incompressible fluid. With the assumption of constant uniform density, Eqs. (12) and (23) are used to obtain

$$\mathscr{L}_2 = -\rho u\left(\frac{2u}{r^3}f + \left(\frac{2u}{c} - 1\right)\frac{1}{r^2}f' + \left(\frac{u}{c} - 1\right)f''\right). \tag{32}$$

Eqs. (13), (14) and (15) are substituted into Eq. (32) which becomes

$$\mathscr{L}_2 = \frac{\rho u^2}{r}\left[\frac{p - p_\infty}{c\rho_\infty} - 2u + \frac{u^2}{2c}\right] + \rho u\left(1 - \frac{u}{c}\right)\frac{\partial u}{\partial t}. \tag{33}$$

Once Eq. (31) has been solved, $\mathscr{L}_2$ is determined from Eq. (33), which is used in Eq. (17) to obtain $\frac{\partial \rho}{\partial t}$.

$\mathscr{L}_2$ describes the entropy at the boundary. An alternative approach is to state that the entropy is constant in the radial direction by setting $\mathscr{L}_2 = 0$. We find that the maximum relative error caused by this second approach is of the order of $10^{-3}\%$.

3.2.3. Prescription of $\mathscr{L}_3$

The NLAA is based on the polar and azimuthal components of velocity being zero on Γ. In a two-dimensional scheme, this may not be the case. However, as the NLAA makes no provision for determining the variation of polar velocities with radius, we make the most basic approximation possible, and state that the variation of polar velocity with radius is zero. Hence, if $u > 0$ on Γ, then $\mathscr{L}_3$ is determined from Eq. (22), otherwise $\mathscr{L}_3 = 0$. This assumption is equivalent to stating that there is no advection of transverse velocities through Γ.

3.2.4. Prescription of $\mathscr{L}_4$

Since the motion on Γ is always subsonic, $\mathscr{L}_4$ is defined by Eq. (25).

4. Computational implementation

Numerical results are obtained by solving the Euler equations on a fixed domain, Ω, as in Section 3. The coordinate system is aligned with the polar axis pointing vertically upwards. In two dimensions with symmetry about the polar axis, the Euler equations may be written as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial r} + \frac{\partial \mathbf{G}(\mathbf{U})}{\partial \theta} + \mathbf{S_r}(\mathbf{U}) + \mathbf{S}_\theta(\mathbf{U}) = \mathbf{D}(\mathbf{U}), \tag{34}$$

where

$$\mathbf{U} = [\,\rho, \quad \rho u, \quad \rho v, \quad E\,]^T, \tag{35}$$

$$\mathbf{F} = \left[\,\rho u, \quad \rho u^2 + p, \quad \rho uv, \quad u(E+p)\,\right]^T \tag{36}$$

and

$$\mathbf{G} = \left[\,\rho v, \quad \rho uv, \quad \rho v^2 + p, \quad v(E+p)\,\right]^T, \tag{37}$$

in which ρ, u, v, E and p are the density, radial velocity, polar velocity, total energy and pressure respectively. The source terms $\mathbf{S_r}$ and $\mathbf{S}_\theta$ are due to the polar coordinate system, and are given by

$$\mathbf{S_r} = \frac{2}{r}\left[\,\rho u, \quad \rho u^2, \quad \rho uv, \quad u(E+p)\,\right]^T \tag{38}$$

and

$$\mathbf{S}_\theta = \frac{1}{r\tan\theta}\left[\,\rho v, \quad \rho uv, \quad \rho v^2, \quad v(E+p)\,\right]^T. \tag{39}$$

The effects of gravity are accounted for by $\mathbf{D}$, defined by

$$\mathbf{D} = \left[\,0, \quad -\rho g\cos\theta, \quad \rho g\sin\theta, \quad -\rho g(u\cos\theta - v\sin\theta)\,\right]^T, \tag{40}$$

where g is gravity, and $g = 9.81\ \mathrm{ms}^{-1}$. The equations are closed with a stiffened gas equation of state given by

$$p = (\gamma - 1)\rho e - \gamma p_c, \tag{41}$$

where $E = \rho e + \frac{1}{2}\rho u^2$. For air we have $\gamma = 1.4$ and $p_c = 0$, which is equivalent to the ideal gas equation of state. For water we use, typically, $\gamma = 7.0$ and $p_c = 3 \times 10^8$ Pa.

Two-phase flow simulations are achieved using a single phase finite volume Euler solver in combination with a ghost fluid method (GFM) to account for the interface.

4.1. Single phase Euler solver

The single-phase Euler solver is a dimensionally-split first order Godunov-type scheme (see, for example, [34,35]). Spatial reconstruction is piecewise constant in each cell based on cell centre values $\mathbf{U}^n_{i,j}$, where i and j denote the spatial indices of the cell in the radial and polar directions respectively, and n denotes the time index. Riemann problems at the cell faces are then defined by $R_{i+\frac{1}{2},j} = R(\mathbf{U}^n_{i,j}, \mathbf{U}^n_{i+1,j})$, and $R_{i,j+\frac{1}{2}} = R(\mathbf{U}^n_{i,j}, \mathbf{U}^n_{i,j+1})$, and are solved using a Roe-average Riemann solver (due to [36]) to obtain HLLC fluxes $\hat{\mathbf{F}}_{i+\frac{1}{2},j}$ and $\hat{\mathbf{F}}_{i,j+\frac{1}{2}}$ (see [37]). Source terms due to the spherical coordinate system and gravity are accounted for using a first-order operator splitting procedure [26].

4.2. Ghost fluid method

Ghost fluid methods are a family of front-tracking methods for the simulation of multimedium flows with sharp interfaces, first developed by Fedkiw [38]. Ghost fluid methods provide a relatively simple way to model multifluid flows with sharp interfaces. We use a variation of the 'real GFM' of [39]. We omit the isobaric fix of [39], and modify only the ghost cells.

The air-water interface is tracked using a level set, ϕ, which is initialised as a signed distance function of the interface and updated according to the advection equation $\phi_t + u_{LS}\phi_r = 0$. Spatial derivatives of the level set are obtained with a weighted essentially non-oscillatory (WENO) spatial reconstruction scheme due to [40]. For one-dimensional simulations, the level set velocity, u_{LS}, is taken to be equal to the velocity of the interface, and hence no re-initialisation procedure is required. For two-dimensional simulations the level set velocity is set as the local fluid velocity, and the level set reinitialisation equation, $\phi_t = \mathrm{sgn}(\phi^0)(1 - |\nabla\phi|)$, where ϕ^0 is the solution to the level set equation before reinitialisation, is solved to retain the signed distance function. We use a scheme due to [41], which is first-order accurate over the whole domain, and second-order accurate in the vicinity of the interface. By construction the interface cannot lie outside the domain. For both level set equations we use a first order scheme for time integration.

4.2.1. Ghost fluid method in one dimension

In one dimension the ghost fluid method is applied as follows. At each time step the location of the interface is determined by finding the zero level set. The index of the cell with cell centre immediately to the left of the interface is denoted q. A two-fluid Riemann problem is constructed, defined by $R(\mathbf{U}^n_q, \mathbf{U}^n_{q+1})$. The Riemann problem is solved using the two-fluid approximate Riemann solver of [36], to provide the left and right star states, denoted $\mathbf{U}^\star_L$ and $\mathbf{U}^\star_R$. The two-fluid domain, Ω is duplicated to create two one-fluid domains, Ω_1 and Ω_2. Ω_1 contains real cells where $\phi \le 0$, $i \le q$, and a band of ghost cells where $\phi > 0$ ($i \ge q+1$). Ω_2 is populated by real cells where $\phi > 0$ and a band of ghost cells where $\phi \le 0$. The band of ghost cells is required to be a minimum thickness of 2 cells for first-order methods, with higher-order methods requiring more ghost cells. Properties in the one-fluid domains are denoted $\mathbf{U}^n_{i,\Omega_1}$ and $\mathbf{U}^n_{i,\Omega_2}$. The cells in the one-fluid domains are populated according to

$$\mathbf{U}^n_{i,\Omega_1} = \begin{cases} \mathbf{U}^n_i & \text{if } i \le q \\ \mathbf{U}^\star_L & \text{if } i \ge q+1 \end{cases} \tag{42}$$

$$\mathbf{U}^n_{i,\Omega_2} = \begin{cases} \mathbf{U}^\star_R & \text{if } i \le q \\ \mathbf{U}^n_i & \text{if } i \ge q+1 \end{cases} \tag{43}$$

The single-phase Euler solver is now used to update each of the domains separately, yielding $\mathbf{U}^{n+1}_{i,\Omega_1}$ and $\mathbf{U}^{n+1}_{i,\Omega_2}$. The level set is updated, then the two one-fluid domains are recombined to the two-fluid domain according to

$$\mathbf{U}^{n+1}_i = \begin{cases} \mathbf{U}^{n+1}_{i,\Omega_1} & \text{if } \phi^{n+1}_i \le 0 \\ \mathbf{U}^{n+1}_{i,\Omega_2} & \text{if } \phi^{n+1}_i > 0 \end{cases} \tag{44}$$

4.2.2. Ghost fluid method in two dimensions

We find that the version of the real GFM used in the one-dimensional simulations does not provide adequate stability as the interface becomes significantly warped in two-dimensional simulations. In two dimensions we use a variation, which is the same as for the one dimensional case, except that whilst densities and velocities are defined by the solution to a Riemann problem, pressures are extrapolated from the region containing air. This modification is based on the modified GFM of [42], and the argument that the motion is predominantly constrained by air pressures and water velocities. A comparison of both versions of ghost fluid method on one-dimensional problems provided similar results. To apply this version of the GFM in two dimensions, we perform the following steps.

1. Determine whether each cell is beside the interface, and if so label it an 'interface cell'. A cell, A, with indices i_A, j_A, is an interface cell if there is a cell, B, such that
$$i_A - 1 \leq i_B \leq i_A + 1, \tag{45}$$
$$j_A - 1 \leq j_B \leq j_A + 1, \tag{46}$$
and
$$\phi_A \phi_B \leq 0. \tag{47}$$
2. For every interface cell, A
   (a) find the partner cell, B, which satisfies Eqs. (45), (46) and (47), and minimises the angle between the level set normals of the two cells, by finding B which minimises $(1 - \nabla\phi_A \cdot \nabla\phi_B)$;
   (b) determine the components of the velocity in directions normal and tangent to the interface, u_n and u_t, where the normal to the interface is positive in the direction from fluid 1 to fluid 2 for cells A and B;
   (c) solve a Riemann problem defined by $R(\tilde{\mathbf{U}}_A^n, \tilde{\mathbf{U}}_B^n)$, where $\tilde{\mathbf{U}} = (\rho, u_n, u_t, p)^T$, obtaining star states $\mathbf{U}_L^\star$ and $\mathbf{U}_R^\star$. Find the components of the star state velocities in the radial and polar directions;
   (d) use the star states of the Riemann problem to define the densities and velocities in the ghost cell, B, beside the interface. Copy the air pressures from the interface cell in the air region.
3. Extrapolate the primitive properties away from the interface in the ghost regions by advecting with the level set normal:
$$\frac{\partial \xi}{\partial \tau} + \mathrm{sgn}(\phi)\nabla\phi \cdot \nabla\xi = 0, \tag{48}$$
for $\xi = \rho, u, v, p$. Eq. (48) is solved with first order upwind discretisation of spatial derivatives and a first order Euler method for time integration.
4. Update the properties in each domain separately using the single-phase Euler solver, to obtain $\mathbf{U}_{\Omega_1}^{n+1}$ and $\mathbf{U}_{\Omega_2}^{n+1}$.
5. Update the level set one time step, obtaining ϕ^{n+1}.
6. Reconstruct the properties in the two-fluid domain according to the sign of the level set:
$$\mathbf{U}_{i,j}^{n+1} = \begin{cases} \mathbf{U}_{\Omega_1,i,j}^{n+1} & \text{if } \phi_{i,j}^{n+1} \leq 0 \\ \mathbf{U}_{\Omega_2,i,j}^{n+1} & \text{if } \phi_{i,j}^{n+1} > 0. \end{cases} \tag{49}$$

This completes the time-step.

4.3. Boundary conditions

The NLAA boundary condition is applied by updating a band of ghost cells on Γ using the characteristic boundary condition formalism as described in Section 3 and a first order method for time integration. A reflecting boundary condition is applied at the origin. For the two-dimensional cases a reflecting boundary condition is also applied on the boundaries where $\theta = 0$ and $\theta = \pi$.

4.4. Limitations of the numerical scheme

Our investigations are carried out using a scheme which is first order in time and space. Higher-order schemes including a second-order MUSCL scheme [43] and a 5th-order WENO scheme [40] with third-order time integration [44] were investigated. We have found that a first-order scheme provides best results. We compare our results for a one-dimensional underwater explosion problem with results of other authors [33,45]. We find that our results using a first order scheme closely match those in [33,45], whilst higher-order methods lead to severely damped bubble oscillations. We believe this is due to the different momentum and energy fluxes through the interface when the GFM is used in conjunction with a numerical scheme based on a wide stencil.

A flaw in the current scheme is that the Euler equations in polar coordinates are not in conservative form, due to the geometric source terms. When the motion of the interface is in the radial direction the regions either side of the interface are subject to erroneously high or low energies, and the Rankine–Hugoniot conditions are not met at the interface. The obvious symptom of these errors is a pressure discontinuity at the interface (visible in Fig. 3) which is proportional in magnitude to the radial interface speed and the grid size. This error is reduced by refining the computational mesh. This is an open problem, and one which the authors are currently investigating.

5. Numerical results in one dimension

We test our method on single-phase and two-phase test problems in one dimension. In all cases, the computational domain is defined by $0 \leq r \leq R_D$, and is made up of uniform cells of width δr, and subject to spherical symmetry. Test problems are run with a range of values for R_D and δr. All cases are run with a CFL number of 0.8.

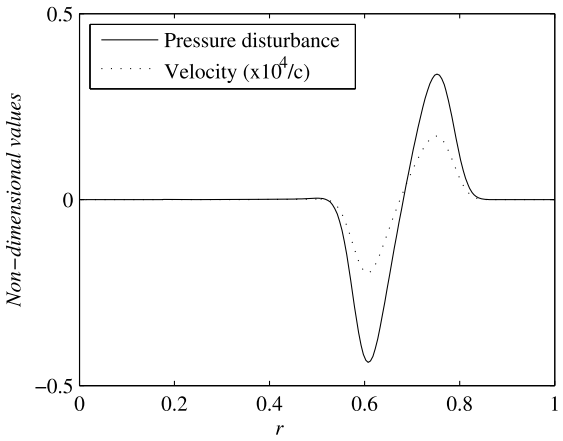


Fig. 1. Problem I: Spatial pressure disturbance and velocity profiles for the outgoing pulse at time $t = 0.005$. Note that the velocity profile has been made non-dimensional with the local speed of sound and magnified by a factor of 10^4.

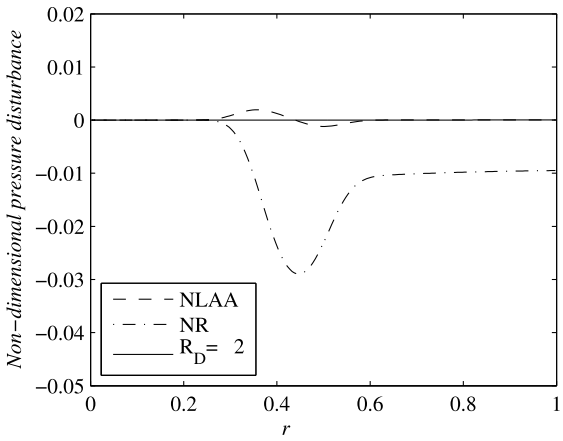


Fig. 2. Problem I: The remaining pressure disturbances after the main pulse has left the domain due to different boundary conditions, at time $t = 0.011$. Solid line: large domain 'ideal' boundary condition. Dashed line: our non-linear acoustic approximation boundary condition. Dot-dashed: Thompson's [9] non-reflecting boundary condition.

For all one-dimensional test problems gravity is neglected and the largest value of R_D is chosen such that there is insufficient time during the simulation for errors caused by the boundary condition to propagate back into the region of interest. This provides us with what is effectively an 'ideal' boundary condition, against which to test the NLAA boundary condition. We refer to these cases as 'large domain' or 'ideal' boundary condition cases.

5.1. Problem I – travelling pulse in water

We first consider a single phase problem. The problem consists of a domain containing water initially at rest, with uniform density. The pressure of a sphere of water near the origin is increased relative to the surrounding water. These initial conditions produce a pulse which propagates outwards at the local speed of sound. Behind the outgoing pulse the velocity is zero and the pressure is uniform. The initial conditions are

$$(\rho,\ u,\ p,\ \gamma,\ p_c) = \begin{cases} (1,\ 0,\ 10,\ 7,\ 3000) & \text{if } 0 < r < 0.1, \\ (1,\ 0,\ 1,\ 7,\ 3000) & \text{if } 0.1 < r < R_D. \end{cases} \tag{50}$$

The simulation is run on a grid with $\delta r = 0.005$ for 400 time steps. We run the simulation on a domain with size $R_D = 1$ for our new artificial boundary condition (NLAA) and also for Thompson's [9] non-reflecting boundary condition (NR). We run the simulation on a domain with $R_D = 2$ to provide an ideal boundary condition. We calculate the pressure disturbance as the relative deviation of the absolute pressure from the initial pressure at the domain boundary (i.e. $(p(r,t) - p_\infty)/p_\infty = p(r,t) - 1$). We make the velocity non-dimensional by multiplying by $10^4/c$, where c is the local speed of sound. Fig. 1 shows the velocity and pressure disturbances due to the outgoing pulse. Fig. 2 shows pressure disturbances, caused by the artificial boundary condition, propagating back towards the origin. It is apparent from Fig. 2 that in this case the NLAA boundary condition outperforms the NR boundary condition: the disturbance which propagates inwards is of much smaller

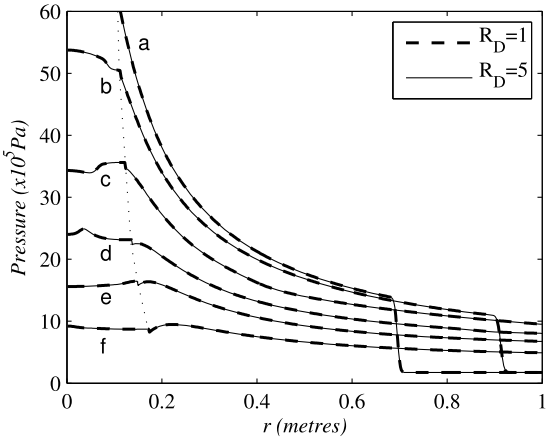


Fig. 3. Problem II-A.1: Spatial pressure profiles at different times for domain sizes of $R_D = 1$ (NLAA BC) and $R_D = 5$ ('ideal' BC). (a) $t = 0.22$ ms; (b) $t = 0.29$ ms; (c) $t = 0.43$ ms; (d) $t = 0.59$ ms; (e) $t = 0.75$ ms; (f) $t = 1.10$ ms. The dotted line shows the location of the interface between air and water.

magnitude than that produced by Thompson's [9] boundary condition. The final pressure reached after a long time has elapsed is correct when using the NLAA boundary condition, but not when using the NR boundary condition. In this case the Mach number of the pulse as it impacts on the domain boundary is very low, at 1.5×10^{-5}. The strength of the pulse leaving the domain is very weak, and this case satisfies the assumptions on which the non-linear acoustic approximation is based. The amplitude of the spurious pulse caused by the boundary condition is approximately 350 times smaller than the amplitude of the pulse which propagates outwards.

5.2. Problem II-A.1 – air gun bubble problem – early stages

We consider the problem of the bubble produced by a seismic air gun. This problem consists of an initially stationary bubble of air at high pressure in water. The initial conditions are:

$$(\rho,\ u,\ p,\ \gamma,\ p_c) = \begin{cases} (102,\ 0,\ 8.85 \times 10^6,\ 1.4,\ 0) & \text{if } 0 < r < 0.1, \\ (1000,\ 0,\ 1.77 \times 10^5,\ 7.0,\ 3 \times 10^8) & \text{if } 0.1 < r < R_D, \end{cases} \tag{51}$$

where all quantities are given in S.I. units. This problem is comparable with an air gun with a volume of 250 cubic inches, charged to a pressure of 2000 pounds per square inch at a depth of 7.7 metres. Air guns of this size and pressure are commonly used in industry. Note that the discrepancy in initial pressures – 8.85×10^6 Pa ≈ 1300 psi – is intentional, and is designed to account for the process by which air is released from the gun. We run the simulation for domain sizes, R_D, of 1 and 5 metres. In both simulations, a grid cell size of $\delta r = 2 \times 10^{-4}$ metres is used. The results with $R_D = 5$ are taken to be the ideal boundary condition case.

Fig. 3 shows the pressure profiles at different times for the two domain sizes. Note the small discontinuity in pressure at the interface due to the ghost fluid method as discussed in Section 4.4. For both domain sizes, the pressure profiles match very closely and cannot be distinguished in Fig. 3. As the outgoing pressure wave passes the domain boundary a disturbance due to the artificial boundary condition forms and propagates back into the domain. This disturbance causes density, velocity and pressure errors of -3×10^{-5}%, 0.007% and -0.04% respectively. That the error in pressure is negative and the error in velocity is positive implies that the artificial boundary condition is applying too weak a resistance at the boundary and is causing higher fluxes at the boundary than in the ideal case.

5.3. Problem II-A.2 – air gun bubble – long run

We now consider the same problem as in the previous case but on longer time scales. We run the simulation for domain sizes, R_D, of 1, 2, 4, 8, 16 and 125 metres, with a cell size of $\delta r = 5 \times 10^{-3}$ metres. The simulation is run for 5×10^4 time steps, which corresponds to approximately 0.14 seconds, during which time the bubble undergoes two full oscillations. During the simulation, the maximum outgoing pressure wave impacts on the boundary in the $R_D = 125$ case; there is insufficient time for any disturbances to propagate back towards the origin as far as $r = 16$. We take the case of $R_D = 125$ as the case with ideal boundary condition with which to compare results obtained on smaller domains.

Fig. 4 shows the time evolution of the interface position, R_{int}, and pressure, P_{int}, for $R_D = 1$. Fig. 5 shows the magnitude in the relative error of the maximum interface position as R_D is varied. The results show third-order convergence of R_{int} with increasing R_D. P_{int} also shows third-order convergence. This convergence fails for $R_D = 16$ and 32, as the variation in density on R_D in these cases is of the same order of magnitude as machine precision.

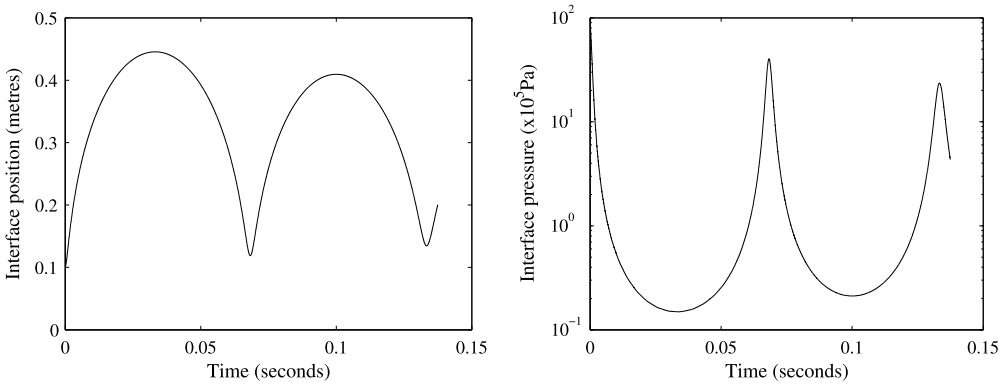


Fig. 4. Problem II-A.2: Interface position and interface pressure variation as a function of time for $R_D = 1$.

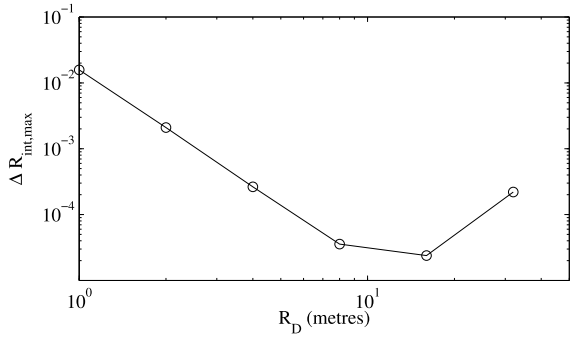


Fig. 5. Problem II-A.2: Variation of the magnitude of the relative error in maximum interface position for different domain sizes. $\Delta R_{int,\max} = |\max(R_{int,R_D}) - \max(R_{int,R_D=125})| / \max(R_{int,R_D=125})$.

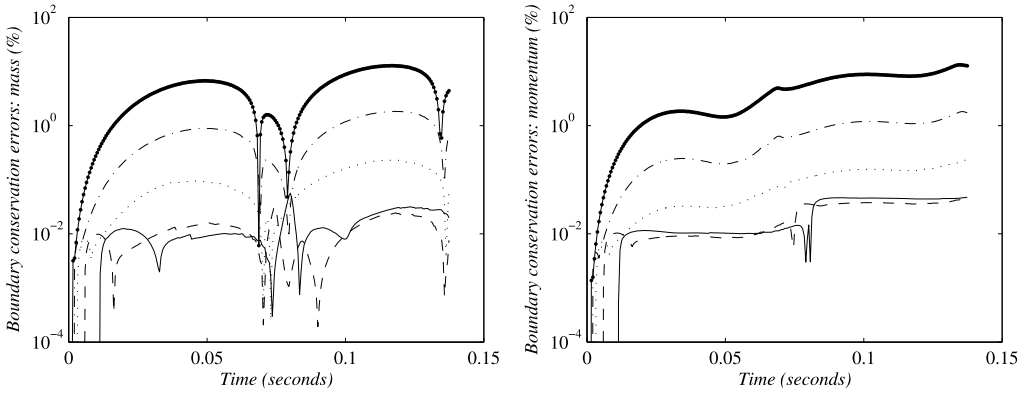


Fig. 6. Problem II-A.2: Relative mass and momentum boundary conservation errors as functions of time for different values of R_D: $R_D = 1$ – solid line with points; $R_D = 2$ – dash-dot line; $R_D = 4$ – dotted line; $R_D = 8$ – dashed line; $R_D = 16$ – solid line.

For each value of R_D, the fluxes of the conservative properties are calculated at the domain boundary. The fluxes are also calculated at the same position for the case of $R_D = 125$. These fluxes are then integrated with respect to time to determine the total quantity of each conserved property which has left the domain. We then determine the relative errors in these cumulative fluxes, taking the case of $R_D = 125$ as a reference. Fig. 6 shows the relative errors in the conservation properties of the boundary for differing domain sizes. Boundary conservation errors in energy match those in mass to within 0.1% in all cases. Fig. 6 shows a maximum error in boundary conservation after two full bubble oscillations (5×10^4 time steps) of approximately 10%. Fig. 6 also shows a third-order convergence in boundary conservation errors as R_D is increased. When $R_D = 16$, variation in density at the domain boundary is of the same order of magnitude as machine precision errors, and hence the convergence characteristics of the smaller domains in Fig. 6 appear not to hold.

This convergence rate is independent of mesh size, although errors due to the boundary conditions are reduced with finer mesh. We run the simulation for two domain sizes, $R_D = 1$ and $R_D = 2$ for a range of grid sizes, $\delta r = 5 \times 10^{-3}$,

Table 1
Problem II-A.2: Relative errors due to boundary conditions and convergence rate with mesh refinement.

δr (metres)	$E_{R_{int}}$	R_c	$E_{P_{int}}$	R_c
5×10^{-3}	7.05×10^{-3}		0.112	
2×10^{-3}	3.08×10^{-3}	0.916	0.047	0.958
1×10^{-3}	1.57×10^{-3}	0.981	0.024	0.991
3.33×10^{-4}	5.53×10^{-4}	0.946	8.3×10^{-3}	0.947

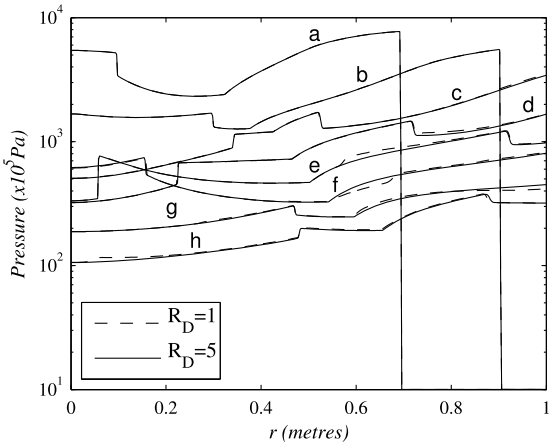


Fig. 7. Problem II-B.1: Spatial pressure profiles at different times for domain sizes of $R_D = 1$ (NLAA BC) and $R_D = 5$ ('ideal' BC). (a) $t = 0.21$ ms; (b) $t = 0.31$ ms; (c) $t = 0.41$ ms; (d) $t = 0.5$ ms; (e) $t = 0.6$ ms; (f) $t = 0.73$ ms; (g) $t = 0.91$ ms; (h) $t = 1.13$ ms.

2×10^{-3}, 1×10^{-3} and 3.33×10^{-3} metres, corresponding to 200, 500, 1000 and 3000 cells per metre respectively. With 3000 cells per metre, the simulation was run for 5×10^5 time steps. The relative error in interface position, $E_{R_{int}}$, and the relative error in interface pressure $E_{P_{int}}$ between the two domain sizes is calculated for each grid size. Table 1 shows the L_1 norm and the convergence rates for these errors. We find approximately first-order convergence of these errors with grid size.

5.4. Problem II-B.1 – gaseous explosion in water – early stages

We consider an underwater explosion problem first studied by Flores and Holt [26]. Other authors have investigated this problem, using both Eulerian [28,32,33] and arbitrary Lagrangian–Eulerian [29–31] methods. This problem is similar to the problem of modelling a seismic air gun, but with a much greater initial pressure. The strength of the propagating shock in this problem is greater than the non-linear acoustic approximation was designed for. As this problem is beyond the remit of the NLAA, it is a good test for the robustness of our method. The initial conditions are

$$(\rho,\ u,\ p,\ \gamma,\ p_c) = \begin{cases} (1.63,\ 0,\ 83810,\ 1.4,\ 0) & \text{if } 0 < r < 0.16, \\ (1.025,\ 0,\ 10,\ 5.5,\ 4921.15) & \text{if } 0.16 < r < R_D. \end{cases} \tag{52}$$

The simulation is run for domain sizes of $R_D = 1$ and $R_D = 5$, with $\delta r = 3.33 \times 10^{-4}$ metres in both cases. The case of $R_D = 5$ provides the ideal boundary conditions. The maximum Mach numbers are 1.07 in the bubble and 0.37 in the water. The maximum Mach number at $r = 1$ is 0.11.

Fig. 7 shows the pressure profile at different times for both $R_D = 1$ and $R_D = 5$. Initially a shock wave propagates are from the interface into the water, and a rarefaction wave propagates into the bubble towards the origin. The shock wave is visible at about $r = 0.7$ in curve (a). As the rarefaction wave impacts on the origin it is reflected as a rarefaction wave, the pressure near the origin drops below the pressure in the rest of the bubble, and an inward propagating shock forms. This shock reflects of the origin and propagates outwards (it is visible at about $r = 0.1$ in curve (a)). Curve (b) shows this shock just prior to impacting on the interface. When it impacts on the interface, it is partially reflected back towards the origin, and partially transmitted out into the water (curve (c)). As the outgoing pressure wave impacts on the boundary a small disturbance forms and propagates back towards the origin (curves (c), (d) and (e)). The disturbance causes maximum errors of −15%, 1.5% and −0.1% in the pressure, velocity and density, respectively, at the boundary. The disturbance propagates in to the air-water interface, at which point it is partially reflected outwards, and partially transmitted into the bubble, where it grows in strength as it converges on the origin (curves (f), (g) and (h)).

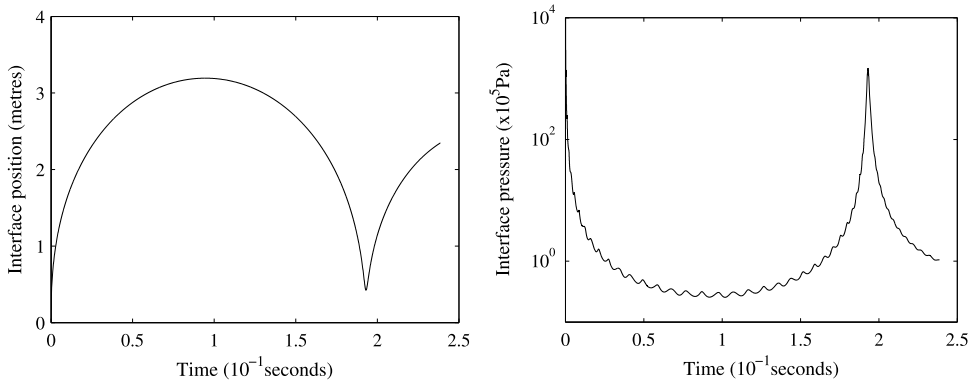


Fig. 8. Problem II-B.2: Interface position and interface pressure variation as a function of time for $R_D = 4$.

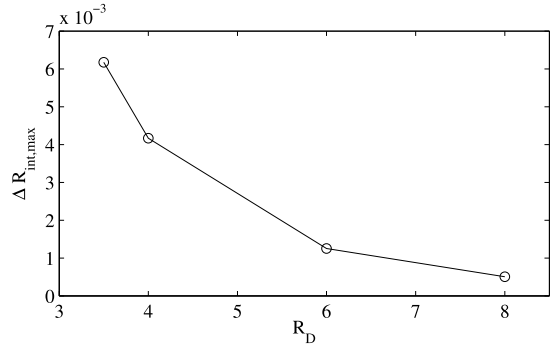


Fig. 9. Problem II-B.2: Variation of the magnitude of the relative error in maximum interface position for different domain sizes. $\Delta R_{int,\max} = |\max(R_{int,R_D}) - \max(R_{int,R_D=125})| / \max(R_{int,R_D=250})$.

5.5. Problem II-B.2 – gaseous explosion in water – long run

We now test our method on the underwater explosion problem over a much greater run time, for one full bubble oscillation period. The initial conditions are the same as for the previous case. We now use a coarser grid, with $\delta r = 4 \times 10^{-3}$ metres. The simulation is run for domain sizes of $R_D = 3.5$, $R_D = 4$, $R_D = 6$, $R_D = 8$ and $R_D = 250$, for 1.25×10^5 time-steps. During the simulation the outgoing pressure wave reaches the boundary in all cases. When $R_D = 250$ spurious reflections from the boundary do not have time to propagate further inwards than $r = 100$. Hence the case of $R_D = 250$ is taken to be the ideal boundary condition with which to compare the performance of the boundary conditions on the smaller domains.

Fig. 8 shows the time-evolution of the bubble radius and the interface pressure. These results are in good agreement with previous authors [29,33]. The secondary oscillations ('internal bubble oscillations') present in the interface pressure in Fig. 8 are due to pressure waves propagating across the bubble and interacting with the air-water interface, as described in the previous section. The maximum radius of the bubble during the simulation is 3.2 metres. The maximum Mach number at the domain boundary when $R_D = 4$ is 0.02, and the initial shock which impacts on the boundary has a pressure ratio of 50. Fig. 9 shows the magnitude of the relative error in maximum interface position for different values of R_D. The results show third-order convergence.

Fig. 10 shows the time evolution of the error in the total mass of air in the bubble. The variation in Fig. 10 of order 1% is due to the non-conservative properties of the ghost fluid method about the interface, and is unavoidable given the current numerical scheme, although it can be reduced with mesh refinement. The differences in mass conservation for different sized domains show no improved performance with larger domains. The variation between the traces in Fig. 10 is due to the sensitivity of the conservation properties of the scheme to the time at which any disturbances from the domain boundary impact on the material interface. It must be noted that the interaction between the disturbance due to the boundary conditions and the wave inside the bubble has the ability to change the phase of the internal bubble oscillations significantly as the bubble collapses. As with previous test problems, we observe third-order convergence of interface position and pressure as R_D is increased.

Fig. 11 shows the conservation errors in mass and momentum flowing out of the domain at R_D relative to the $R_D = 250$ case for each value of $R_D = 4, 6$ and 8. As in the previous cases, boundary conservation errors in energy matched those in mass to within 0.1%. These results show a third-order convergence for boundary conservation properties with increasing R_D.

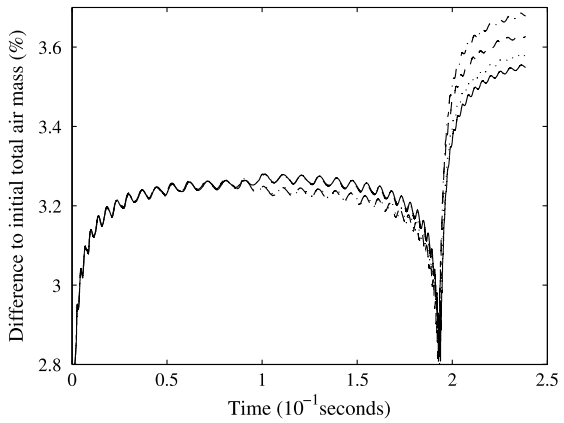


Fig. 10. Problem II-B.2: Variation of air mass as a function of time for different values of R_D. Dashed line – $R_D = 4$; dash-dot line – $R_D = 6$; dotted line – $R_D = 8$; solid line – $R_D = 250$.

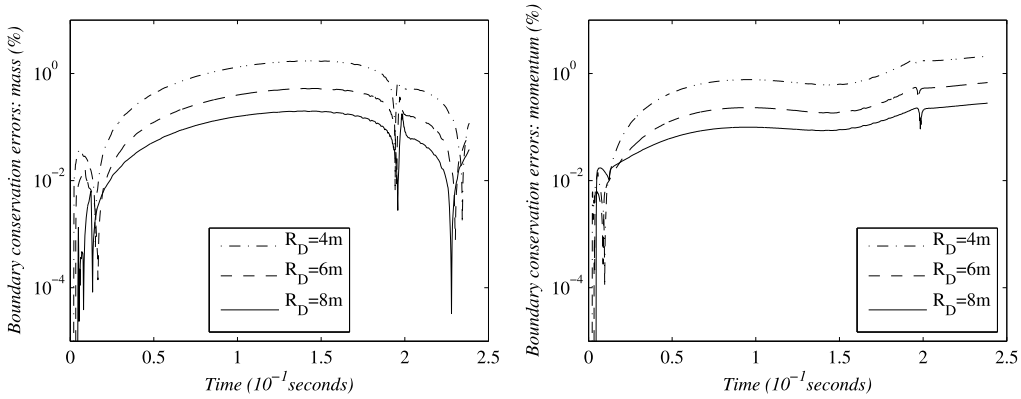


Fig. 11. Problem II-B.2: Relative mass and momentum boundary conservation errors as a function of time for different values of R_D.

For the case of $R_D = 8$, the maximum errors in boundary conservation are of the order of 0.1%. This is a very good conservation property, given that the simulation has been run for such a large number of time steps.

We also run the simulation with $R_D = 3.21$, in which case R_D is 0.1% greater than the maximum bubble radius. In this case the results obtained match those expected from the convergence properties observed above. Recall that the simulation will break down if the interface moves outside the domain. We note that for the underwater explosion problem the minimum acceptable domain size is determined not by the performance of the boundary condition, but by the requirement that the domain is larger than the maximum bubble radius.

The NLAA boundary condition provides excellent results with regard to the large-scale motion of the bubble. Smaller scale motion, such as the pressure waves oscillating within the bubble, are significantly affected by the boundary conditions, but they themselves have little effect on the bubble radius or interface pressure.

6. Results in two-dimensions

In our two-dimensional test problems, the computational domain is the region containing the points $0 \leq r \leq R_D$ and $0 \leq \theta \leq \pi$, and is split uniformly into cells with side lengths δr and $r\delta\theta$, where $\delta r = R_D/50$ and $\delta\theta = \pi/50$. In all cases, we set $R_D = 1$ and use a CFL number of 0.8.

6.1. Problem III-A.1 – two-dimensional air gun bubble subject to a disturbance

We now simulate an air gun bubble as in Problem II-A, but the imposition of spherical symmetry is relaxed and the initial shape is subject to a small sinusoidal disturbance, η. For this problem we neglect gravity. The initial conditions are given by

$$(\rho,\ u,\ p,\ \gamma,\ p_c) = \begin{cases} (102,\ 0,\ 8.85 \times 10^6,\ 1.4,\ 0) & \text{if } 0 < r < 0.1 + \eta, \\ (1000,\ 0,\ 1.77 \times 10^5,\ 7.0,\ 3 \times 10^8) & \text{if } 0.1 + \eta < r < R_D, \end{cases} \tag{53}$$

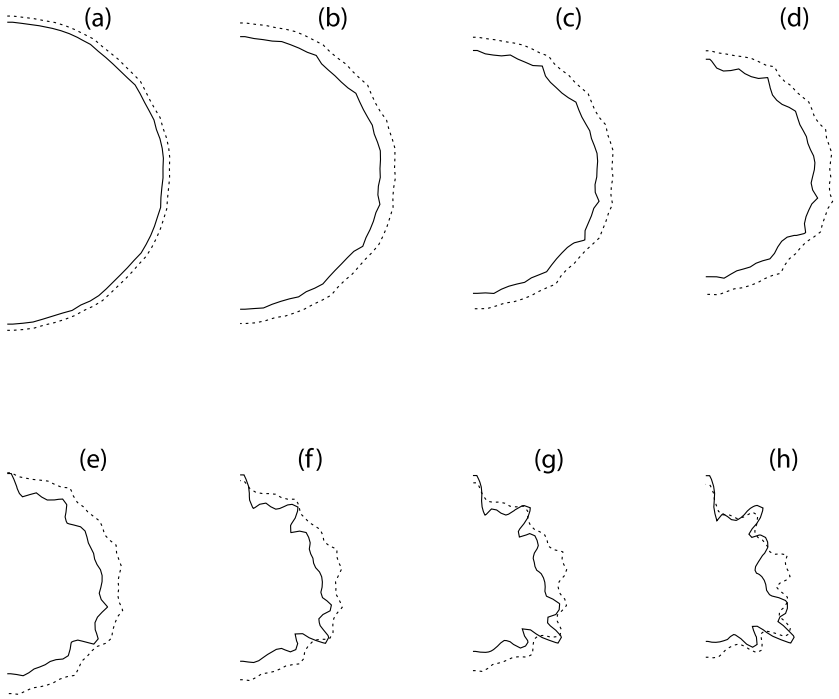


Fig. 12. The shape of the bubble at different times during collapse, with $R_D = 1$. Gravity is neglected. The initial shape of the bubble is a sphere subject to a sinusoidal disturbance, $\eta = 0.001\sin(20\theta)$. The solid line shows results with $R_D = 1$. The dashed line shows results with $R_D = 2$. (a) $t = 29.2$ ms; (b) $t = 50.2$ ms; (c) $t = 56.7$ ms; (d) $t = 61.5$ ms; (e) $t = 64.7$ ms; (f) $t = 67.2$ ms; (g) $t = 69.0$ ms; (h) $t = 70.8$ ms.

where all quantities are given in S.I. units. We set $\eta = 0.001\sin(20\theta)$. Note that if we set $\eta = 0$ the initial conditions do not vary with θ, and the problem collapses to the one-dimensional problem. We run the simulation twice, with $R_D = 1$ and $R_D = 2$.

Fig. 12 shows the shape of the bubble at different times during collapse, for the case of $R_D = 1$ (solid line) and $R_D = 2$ (dashed line). As the bubble expands outwards the disturbances to the interface do not grow, but are damped slightly, and so we do not show the shape of the bubble during the expansion phase. As the bubble collapses, the interface becomes unstable and the disturbances grow. This instability is a expression of Rayleigh–Taylor instability [46]. Our results agree with [47,48], in which the changing form of the bubble surface during collapse can be seen in high-speed photographs of an air gun bubble. Due to the instability of the surface, the final shape of the bubble is highly sensitive to small changes during the earlier stages of oscillation. In Fig. 12 the initial bubble shapes for the two cases are identical, whilst the final shapes of the two bubbles differ significantly. This is due to the small differences caused by errors due to the boundary condition. If the model were used to study bubble surface instabilities, R_D, δr and $\delta\theta$ should be set such that grid converged results in terms of R_{int} and P_{int} are obtained.

Polar coordinate systems contain a singularity at the poles. It is well-known [49–51] that this singularity causes Rayleigh–Taylor instabilities to grow faster at the poles. This phenomenon is a numerical artifact of the discretisation. We observe this phenomenon in Fig. 12, frames (f) to (h), where a long Rayleigh–Taylor finger protruding along the polar axis is clearly visible.

6.2. Problem III-A.2 – two-dimensional air gun bubble under the influence of gravity

This final problem has the same initial conditions as problem II-A, but the imposition of spherical symmetry is relaxed, and the problem is subject to axi-symmetry about the polar axis. The initial conditions are adjusted to include the effects of gravity by augmenting pressure terms with the hydrostatic pressure, $p_{hydrostatic} = -\rho g r\cos\theta$. An effective bubble radius is obtained by calculating the volume of the bubble, and finding the radius of a sphere of that volume. The interface pressure is taken as the average pressure over the bubble surface.

Fig. 13 shows the shape of the bubble as it collapses, both with (solid line) and without (dashed line) gravity. The case without gravity is spherically symmetric, and the bubble does not undergo any translation. We observe the bubble rising due to gravity, at a rate which is in agreement with previous numerical simulations [25]. As the bubble collapses, our results show a jet begins to form on the underside and pierce upwards through the bubble along the polar axis. This phenomenon is well known, and has been captured previously, for instance in [25,52]. For most of the oscillation (the expansion phase,

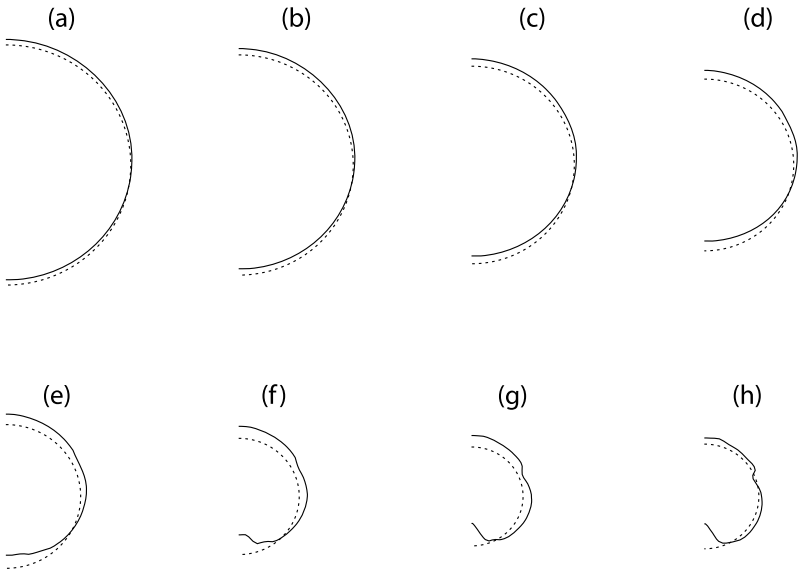


Fig. 13. The shape and position of the bubble at different times during collapse, with $R_D = 1$, both with (solid line) and without (dotted line) gravity included. (a) $t = 56.8$ ms; (b) $t = 59.8$ ms; (c) $t = 62.7$ ms; (d) $t = 65.3$ ms; (e) $t = 67.7$ ms; (f) $t = 69.9$ ms; (g) $t = 71.8$ ms; (h) $t = 73.6$ ms.

and frames (a) to (d) in Fig. 13) the effects of gravity on the effective bubble radius and the bubble pressure are negligible. It is only during the latter stages of collapse that the translation and deformation of the bubble has a significant effect on the pressure far from the bubble. As discussed above, there is a singularity at the poles which increases the speed of growth of instabilities. In long run time simulations, these errors can evolve into axial jets [51]. We find that as the physical jet directed radially inwards approaches the origin, the unphysical jet forms in the opposite direction, and eventually leads to the breakdown of the simulation.

In its current form, the code would take an inordinately long time to simulate one oscillation of a two-dimensional bubble on a large domain, as we did for the one-dimensional results in Section 5. Hence we do not have an 'ideal boundary condition' result to allow the calculation of the absolute errors introduced by the boundary condition in two dimensions. Whilst much of the above discussion is unrelated to our boundary condition, this only serves to highlight its efficacy. The boundary condition allows two-dimensional simulations of oscillating bubbles on small domains, at a reduced computational cost. This reduction in cost can facilitate research into more interesting details of bubble motion, such as surface instabilities and the formation of jets.

7. Conclusions

We have derived a new artificial boundary condition for numerical simulations of oscillating bubbles and similar problems on a finite domain. The method is applicable when the problem is spherical in nature and close to spherically symmetric, and the motion at the domain boundary is of low Mach number (less than 0.1). The boundary condition is based on the non-linear acoustic approximation, developed for use in modelling seismic air guns. We use the non-linear acoustic approximation to calculate an approximate solution to the motion outside the domain based on the solution at the domain boundary. We apply boundary conditions by using the approximate solution to describe all characteristic waves incoming to the domain at the boundary. We implement our boundary condition in one- and two-dimensional two-phase Euler solvers. A Godunov-type scheme is used for single phase calculations, whilst the interface between phases is modelled using a ghost fluid method. The scheme is first-order accurate in space and time. We have tested our method on a range of one- and two-phase problems in one and two dimensions.

In one dimension, the method performs well, yielding accurate results for underwater explosion problems, even when the domain boundary is only slightly larger (0.1%) than the maximum bubble radius. A major benefit of the method is that it allows long run time (10^5 time steps with a CFL number of 0.8) simulations of such problems on a highly truncated domain, at a reduced computational cost. The method is robust, and capable of yielding good conservation properties (errors of less than 1%) over very long run times. Our two-dimensional results show that the boundary condition allows long run time simulations of axisymmetric oscillating bubbles provided computational domain extends some distance into the water such that the motion at the boundary is close to spherically symmetric. The value of our boundary condition to two-dimensional simulations is significant, as it permits complex aspects of bubble behaviour to be simulated at very small computational costs.

Acknowledgements

J.R.C. King is fully supported by Petroleum Geo-Services (PGS). A.M. Ziolkowski is supported by Petroleum Geo-Services and the Royal Academy of Engineering.

References

[1] D. Givoli, Non-reflecting boundary conditions, J. Comput. Phys. 94 (1991) 1–29, http://dx.doi.org/10.1016/0021-9991(91)90135-8.
[2] T. Hagstrom, Radiation boundary conditions for the numerical simulation of waves, Acta Numer. 8 (1999) 47–106, http://dx.doi.org/10.1017/S0962492900002890.
[3] S.V. Tsynkov, Numerical solution of problems on unbounded domains. A review, Appl. Numer. Math. 27 (1998) 465–532, http://dx.doi.org/10.1016/S0168-9274(98)00025-7.
[4] J.-P. Berenger, A perfectly matched layer for the absorption of electromagnetic waves, J. Comput. Phys. 114 (1994) 185–200, http://dx.doi.org/10.1006/jcph.1994.1159.
[5] J.-P. Berenger, Three-dimensional perfectly matched layer for the absorption of electromagnetic waves, J. Comput. Phys. 127 (1996) 363–379, http://dx.doi.org/10.1006/jcph.1996.0181.
[6] M.E. Hayder, F.Q. Hu, M.Y. Hussaini, Towards perfectly absorbing boundary conditions for Euler equations, AIAA J. 37 (1999) 912–918, http://dx.doi.org/10.2514/2.810.
[7] T. Colonius, Modelling artificial boundary conditions for compressible flow, Annu. Rev. Fluid Mech. 36 (2004) 315–345, http://dx.doi.org/10.1146/annurev.fluid.36.050802.121930.
[8] G.W. Hedstrom, Nonreflecting boundary conditions for nonlinear hyperbolic systems, J. Comput. Phys. 30 (1979) 222–237, http://dx.doi.org/10.1016/0021-9991(79)90100-1.
[9] K.W. Thompson, Time dependent boundary conditions for hyperbolic systems, II, J. Comput. Phys. 89 (1990) 439–461, http://dx.doi.org/10.1016/0021-9991(90)90152-Q.
[10] K.W. Thompson, Time dependent boundary conditions for hyperbolic systems, J. Comput. Phys. 68 (1987) 1–24, http://dx.doi.org/10.1016/0021-9991(87)90041-6.
[11] J.W.S. Rayleigh, On the pressure developed in a liquid during the collapse of a spherical cavity, Philos. Mag. 34 (1917) 94–99.
[12] H. Lamb, The early stages of submarine explosion, Philos. Mag. 45 (1923) 257–265.
[13] C. Herring, Theory of the Pulsations of the Gas Bubble Produced by an Underwater Explosion, Office of Naval Research, Dept. of the Navy, 1941.
[14] J.B. Keller, I.I. Kolodner, Damping of underwater explosion bubble oscillations, J. Appl. Phys. 27 (1956) 1152–1161, http://dx.doi.org/10.1063/1.1722221.
[15] F.R. Gilmore, Collapse of a spherical bubble, Tech. Rep. No. 26-4, Hydrodyn. Lab., Calif. Inst. Tech., 1952, http://resolver.caltech.edu/CaltechAUTHORS:Gilmore_fr_26-4.
[16] M.S. Plesset, S.A. Zwick, A nonsteady heat diffusion problem with spherical symmetry, J. Appl. Phys. 23 (1952) 95–98, http://dx.doi.org/10.1063/1.1701985.
[17] A. Prosperetti, M.S. Plesset, Vapour-bubble growth in a superheated liquid, J. Fluid Mech. 85 (1978) 349–368, http://dx.doi.org/10.1017/S0022112078000671.
[18] A. Prosperetti, A. Lezzi, Bubble dynamics in a compressible liquid. Part 1. First-order theory, J. Fluid Mech. 168 (1986) 457–478, http://dx.doi.org/10.1017/S0022112086000460.
[19] A. Lezzi, A. Prosperetti, Bubble dynamics in a compressible liquid. Part 2. Second-order theory, J. Fluid Mech. 185 (1987) 289–321, http://dx.doi.org/10.1017/S0022112087003185.
[20] A. Ziolkowski, A method for calculating the output pressure waveform from an air gun, Geophys. J. R. Astron. Soc. 21 (1970) 137–161, http://dx.doi.org/10.1111/j.1365-246X.1970.tb01773.x.
[21] A. Ziolkowski, Measurement of air-gun bubble oscillations, Geophysics 63 (1998) 2009–2024, http://dx.doi.org/10.1190/1.1444494.
[22] A. Kucera, J.R. Blake, Approximate methods for modelling cavitation bubbles near boundaries, Bull. Aust. Math. Soc. 41 (1990) 1–44, http://dx.doi.org/10.1017/S0004972700017834.
[23] M.C. Hooton, J.R. Blake, W.K. Soh, Behaviour of an underwater explosion bubble near a rigid boundary: theory and experiment, Fluid Dyn. Appl. 23 (1994) 421–428, http://dx.doi.org/10.1007/978-94-011-0938-3_40.
[24] J.R. Blake, G.S. Keen, R.P. Tong, M. Wilson, Acoustic cavitation: the fluid dynamics of non-spherical bubbles, Philos. Trans. R. Soc. Lond. A 357 (1999) 251–267, http://dx.doi.org/10.1098/rsta.1999.0326.
[25] E. Cox, A. Pearson, J.R. Blake, S.R. Otto, Comparison of methods for modelling the behaviour of bubbles produced by marine seismic airguns, Geophys. Prospect. 52 (2004) 451–477, http://dx.doi.org/10.1111/j.1365-2478.2004.00425.x.
[26] J. Flores, M. Holt, Glimm's method applied to underwater explosions, J. Comput. Phys. 44 (1981) 377–387, http://dx.doi.org/10.1016/0021-9991(81)90058-9.
[27] M. Holt, Underwater explosions, Annu. Rev. Fluid Mech. 9 (1977) 187–214, http://dx.doi.org/10.1146/annurev.fl.09.010177.001155.
[28] J.P. Cocchi, R. Saurel, J.C. Loraud, Treatment of interface problems with Godunov-type schemes, Shock Waves 5 (1996) 347–357, http://dx.doi.org/10.1007/BF02434010.
[29] A.R. Pishevar, R. Amirifar, An adaptive ALE method for underwater explosion simulations including cavitation, Shock Waves 20 (2010) 425–439, http://dx.doi.org/10.1007/s00193-010-0275-x.
[30] G. Barras, M. Souli, N. Aquelet, N. Couty, Numerical simulation of underwater explosions using an ALE method. The pulsating bubble phenomena, Ocean Eng. 41 (2012) 53–66, http://dx.doi.org/10.1016/j.oceaneng.2011.12.015.
[31] R.W. Smith, AUSM(ALE): A geometrically conservative arbitrary Lagrangian–Eulerian flux splitting scheme, J. Comput. Phys. 150 (1999) 268–286, http://dx.doi.org/10.1006/jcph.1998.6180.
[32] T.G. Liu, B.C. Khoo, K.S. Yeo, The simulation of compressible multi-medium flow. I. A new methodology with test applications to 1D gas–gas and gas–water cases, Comput. Fluids 30 (2001) 291–314, http://dx.doi.org/10.1016/S0045-7930(00)00022-0.
[33] X.Y. Hu, B.C. Khoo, N.A. Adams, F.L. Huang, A conservative interface method for compressible flows, J. Comput. Phys. 219 (2006) 553–578, http://dx.doi.org/10.1016/j.jcp.2006.04.001.
[34] B. Einfeldt, On Godunov-type methods for gas dynamics, SIAM J. Numer. Anal. 25 (1988) 294–318, http://www.jstor.org/stable/2157317.
[35] M.J. Ivings, D.M. Causon, E.F. Toro, On Riemann solvers for compressible liquids, Int. J. Numer. Methods Fluids 28 (1998) 395–418, http://dx.doi.org/10.1002/(SICI)1097-0363(19980915)28:3<395::AID-FLD718>3.0.CO;2-S.
[36] X.Y. Hu, N.A. Adams, G. Iaccarino, On the HLLC Riemann solver for interface interaction in compressible multi-fluid flow, J. Comput. Phys. 228 (2009) 6572–6589, http://dx.doi.org/10.1016/j.jcp.2009.06.002.
[37] E.F. Toro, M. Spruce, M. Speares, Restoration of the contact surface in the HLL–Riemann solver, Shock Waves 4 (1994) 25–34, http://dx.doi.org/10.1007/BF01414629.

[38] R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows – the ghost fluid method, J. Comput. Phys. 152 (1999) 457–492, http://dx.doi.org/10.1006/jcph.1999.6236.
[39] W. Wang, T.G. Liu, B.C. Khoo, A real ghost fluid method for the simulation of multimedium compressible flow, SIAM J. Sci. Comput. 28 (2006) 278–302, http://dx.doi.org/10.1137/030601363.
[40] R. Borges, M. Carmona, B. Costa, W.S. Don, An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws, J. Comput. Phys. 227 (2008) 3191–3211, http://dx.doi.org/10.1016/j.jcp.2007.11.038.
[41] G. Russo, P. Smereka, A remark on computing distance functions, J. Comput. Phys. 163 (2000) 51–67, http://dx.doi.org/10.1006/jcph.2000.6553.
[42] R.P. Fedkiw, Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method, J. Comput. Phys. 175 (2002) 200–224, http://dx.doi.org/10.1006/jcph.2001.6935.
[43] P. Colella, A direct Eulerian MUSCL scheme for gas dynamics, SIAM J. Sci. Stat. Comput. 6 (1985) 104–117, http://dx.doi.org/10.1137/0906009.
[44] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock capturing schemes, J. Comput. Phys. 77 (1988) 439–741, http://dx.doi.org/10.1016/0021-9991(88)90177-5.
[45] H. Luo, J.D. Baum, R. Löhner, On the computation of multi-material flows using ALE formulation, J. Comput. Phys. 194 (2004) 304–328, http://dx.doi.org/10.1016/j.jcp.2003.09.026.
[46] G. Taylor, The instability of liquid surfaces when accelerated in a direction perpendicular to their planes. I, Proc. R. Soc. Lond. Ser. A 201 (1950) 192–196, http://dx.doi.org/10.1098/rspa.1950.0052.
[47] J. Langhammer, M. Landrø, High-speed photography of the bubble generated by an airgun, Geophys. Prospect. 44 (1996) 153–172, http://dx.doi.org/10.1111/j.1365-2478.1996.tb00143.x.
[48] J. Langhammer, S. Graciet, I. Vik, M. Espeland, O.J. Lokberg, Holographic studies of the bubble generated by a seismic airgun, J. Acoust. Soc. Am. 97 (1995) 362–369, http://dx.doi.org/10.1121/1.412321.
[49] J. Kane, D. Arnett, B.A. Remington, S.G. Glendinning, G. Bazan, E. Muller, B.A. Fryxell, R. Teyssier, Two-dimensional versus three-dimensional supernova hydrodynamic instability growth, Astrophys. J. 528 (2000) 989–994, http://dx.doi.org/10.1086/308220.
[50] K. Kifonidis, T. Plewa, H.T. Janka, E. Muller, Non-spherical core collapse supernovae I. Neutrino-driven convection, Rayleigh–Taylor instabilities, and the formation and propagation of metal clumps, Astron. Astrophys. 408 (2003) 621–649, http://dx.doi.org/10.1051/0004-6361:20030863.
[51] K. Kifonidis, T. Plewa, L. Scheck, H.-T. Janka, E. Muller, Non-spherical core collapse supernovae II. The late-time evolution of globally anistropic neutrino-driven explosions and their implications for SN 1987 A, Astron. Astrophys. 453 (2006) 661–678, http://dx.doi.org/10.1051/0004-6361:20054512.
[52] A.M. Zhang, S.P. Wang, G.X. Wu, Simulation of bubble motion in a compressible liquid based on the three dimensional wave equation, Eng. Anal. Bound. Elem. 37 (2013) 1179–1188, http://dx.doi.org/10.1016/j.enganabound.2013.04.013.

Viscosity in air-gun bubble modelling

Jack King*

* *The Grant Institute,*

School of GeoSciences,

University of Edinburgh,

James Hutton Road,

Edinburgh,

EH9 3FE,

UK

(March 31, 2015)

Running head:

ABSTRACT

I present a finite volume simulation of an air gun bubble in which the compressible Navier-Stokes equations are solved numerically. These equations include viscosity. The viscosity has an effect opposite to that predicted by previous models, and in agreement with published experimental observations. The influence of the value of viscosity on air gun signals is weak, but the effect of including the viscosity at the bubble surface through the no-slip condition is significant. The no-slip condition causes boundary layers at the bubble surface and changes in the velocity structure throughout the bubble. The application of the no-slip condition at the bubble surface causes a reduction of about 20% in the bubble rise rate, as the model captures the effects of skin friction drag on the bubble. The influence of skin friction drag on

bubble rise rate is large enough that it should not be neglected from rise rate calculations.

INTRODUCTION

In order to achieve accurate deconvolution of the seismic data in marine sesimic exploration, it is necessary to have an accurate knowledge of the signal produced by the air gun source or air-gun array. Industry uses modelling to determine this signal, and also in the design of arrays.

Air-gun bubbles were first modelled by Ziolkowski (1970), who used a scheme developed by Gilmore (1952), following from a line of work on cavitation and underwater explosions leading back through Kirkwood and Bethe (1942) and Lamb (1923) to Rayleigh (1917). This same line of work led to the widely used Rayleigh-Plesset equation (Plesset, 1949; Prosperetti and Lezzi, 1986; Lezzi and Prosperetti, 1987). Air-gun models were further developed by, for instance, Dragoset (1984), Laws et al. (1990), Landrø (1992) and Landrø and Sollie (1992), based on the same approximations as Ziolkowski (1970), but with viscous terms added to the equation of motion of the bubble, to provide more damped results. Ziolkowski and Metselaar (1984) presented an improvement to the model of Ziolkowski (1970) which aimed to capture the thermodynamics of the bubble more realistically. Ziolkowski (1998) presented a method to calculate the wavefield produced by an air gun from near-field measurements. The method contains an improvement to the equation of motion used in the model of Ziolkowski (1970), which removes the need for a complex transition between the near-field and far-field. All these air gun models are based on the assumptions of homogeneous spherical bubbles. Cox et al. (2004) reviewed several different air gun bubble models, including one using boundary element methods similar to Blake et al. (1999). In these models the bubbles are not constrained to be spherical, but can deform due to gravity. King et al. (2015) developed an artificial boundary condition for use in finite volume simulations

of oscillating bubbles, with a focus on air gun bubbles. King et al. (2015) demonstrated their method in one and two dimensions, showing the results of long run time axisymmetric fixed-grid finite-volume simulations of oscillating bubbles on highly truncated domains. All the spherically symmetric models referred to above neglect the kinetic energy of the contents of the bubble, and hence neglect any viscous dissipation of energy within the bubble.

The viscous damping terms included by Dragoset (1984); Laws et al. (1990); Landrø (1992), which were based on the approach of Bornhorst and Hatsopoulos (1967) for cavitation bubbles, were introduced to provide a better match with data, at the expense of the integrity of the physics of the model. It was known that air gun bubble models lacked damping when compared with pressure measurements of air-gun bubble oscillations (Ziolkowski, 1970). Whilst viscosity was neglected in the derivation of the equation of motion used in these models, viscous damping terms were appended to the equation in order to obtain a more accurate model. The work of Bornhorst and Hatsopoulos (1967), on which this modification was based, related to cavitation bubbles. Cavitation bubbles are very different from air-gun bubbles, both in terms of scales of motion, and the flow regimes which occur. Langhammer and Landrø (1993) conducted an experiment in which an air gun was fired in a tank whilst the viscosity of the water was changed. This experiment demonstrated that the viscous terms of Dragoset (1984); Laws et al. (1990); Landrø (1992) were unrealistic. The effect of increasing viscosity observed by Langhammer and Landrø (1993) was a reduction in damping, whilst the effect of increasing the viscous damping terms in the models was an increase in damping. The experiment conducted by Langhammer and Landrø (1993) showed that viscosity could not be the cause of the observed damping. The so-called viscous terms in the model did not realistically capture the physical processes by which viscosity influences the bubble motion.

The spherically symmetric models referred to above are based on approximate equations of motion for the water. These approximate equations of motion are founded on a set of assumptions which are not strictly valid at the surface of the bubble. The application of these equations of motion at the bubble surface constrains the models to assume the bubble to be spherical. King et al. (2015) developed a novel artificial boundary condition based on the non-linear acoustic approximation (NLAA) of Ziolkowski (1998). They used this boundary condition in an axisymmetric finite volume model of an air gun bubble in water, in a spherical domain that is centred on the air gun and larger than the maximum volume of the bubble. The radius of the domain is typically about twice the maximum bubble radius. The boundary condition mimics the motion of the water outwards from the boundary to infinity, and allows such finite volume simulations at reduced computational costs. Note that in King et al. (2015) the sea surface was neglected, as in this paper. Whilst this model still contains assumptions about the bubble - axisymmetry is assumed, and viscosity is neglected - the approach does not constrain the model to make these assumptions. The boundary condition is applied around a domain large enough that the assumptions on which the boundary condition is based are valid on the boundary. Any set of assumptions may be adhered to by the model within the computational domain.

The most complex aspect of a finite volume simulation of an air-gun bubble is the air-water interface, which is a moving and distorting discontinuity. Jumps in conservative properties across the interface may be of several orders of magnitude. The bubble motion is slow compared with the sound speed, and hence the simulation must be run for many (often $\mathcal{O}\left(10^5\right)$) time-steps and it is important to use an accurate and non-diffusive technique to model the interface. King et al. (2015) use a technique called a ghost fluid method (GFM) (Fedkiw et al., 1999), in which the interface is tracked, and the governing equations are

solved separately for the air and water, whilst the correct fluxes through the interface are achieved by applying appropriate boundary conditions for each phase. Ghost fluid methods are relatively simple to implement, maintaining a sharp interface, and have been shown to be accurate for air-water interfaces when compared with experimental and theoretical results (Liu et al., 2005; Terashima and Tryggvason, 2009). The topic of interface modelling for multiphase flows is an open one, and one on which many researchers are currently working.

In this paper I present an extension to the scheme in King et al. (2015) in which viscosity is included in the finite volume model. I present numerical results which agree with the experimental observations of Langhammer and Landrø (1993), and show the difference in structure of the bubble with and without viscosity. The results show a reduction in rise rate for the viscous model.

The remainder of this paper is set out as follows. First I describe the computational domain, the governing equations and the numerical method. Then follows Numerical Results, in which the results of the model are presented and discussed. Finally I draw conclusions on the numerical results, which suggest that viscosity influences bubble rise rates.

NUMERICAL METHOD

Consider a Cartesian coordinate system, with origin located on an air gun a depth d below the sea surface, and z increasing with decreasing depth. The gun is assumed to be stationary in still water, and hence I make the approximation that all motion is subject to axisymmetry about the z-axis. Consider next a polar coordinate system with the same origin such that the position vector $\mathbf{r} = (r, \theta)$ in the polar coordinate system is defined by $r = \left(x^2 + y^2 + z^2\right)^{\frac{1}{2}}$ and $\cos\theta = z/r$.

Let Ω be a spherical domain with radius R_D. The boundary of Ω is denoted by Γ. Ω is centred on the origin, and R_D is large enough that the bubble produced by the air gun remains completely within Ω during the period of interest. This configuration is shown in Figure 1. For simplicity, I neglect the physical body of the gun, gun housing and all cables, and consider only the bubble produced when the gun is fired. The governing equations for viscous compressible fluids are the Navier-Stokes equations, which can be expressed as the Euler equations for compressible flow with the addition of source terms for the viscous forces. Within Ω, I solve the compressible Navier-Stokes equations numerically. I apply the non-linear acoustic approximation (NLAA) artificial boundary condition of King et al. (2015) on Γ. Whilst Ω is spherical, the constraint of polar axisymmetry limits the region of computation to the set of points $\mathbf{r} \in ([0, R_D], [0, \pi])$. The domain is discretised into $m \times n$ cells, with side length and divergence angle δr and $\delta\theta$ respectively.

The Navier-Stokes equations in two-dimensional polar coordinates may be written

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial r} + \frac{\partial \mathbf{G}}{\partial \theta} + \mathbf{S_r} + \mathbf{S}_\theta = \mathbf{D} + \mathbf{F}_\mu, \tag{1}$$

where the vector of conserved properties is

$$\mathbf{U} = \left[\begin{array}{cccc} \rho, & \rho u, & \rho v, & E \end{array} \right]^T, \tag{2}$$

the fluxes of $\mathbf{U}$ are

$$\mathbf{F} = \left[\begin{array}{cccc} \rho u, & \rho u^2 + p, & \rho uv, & u\left(E + p\right) \end{array} \right]^T, \tag{3a}$$

$$\mathbf{G} = \left[\begin{array}{cccc} \rho v, & \rho uv, & \rho v^2 + p, & v\left(E + p\right) \end{array} \right]^T, \tag{3b}$$

the geometric source terms are

$$\mathbf{S_r} = \frac{2}{r} \left[\begin{array}{cccc} \rho u, & \rho u^2, & \rho uv, & u\left(E + p\right) \end{array} \right]^T, \tag{4a}$$

$$\mathbf{S}_\theta = \frac{1}{r \tan\theta} \left[\begin{array}{cccc} \rho v, & \rho uv, & \rho v^2, & v\left(E + p\right) \end{array} \right]^T, \tag{4b}$$

and the source terms due to gravity are

$$\mathbf{D} = \left[\begin{array}{cccc} 0, & -\rho g \cos\theta, & \rho g \sin\theta, & \rho g \left(u \cos\theta - v \sin\theta \right) \end{array} \right]^T. \tag{5}$$

The symbols ρ, u, v, E and p are the density, radial and polar velocity components, total energy and pressure, respectively, whilst g is the acceleration due to gravity. Equation 1 is closed with a stiffened gas equation of state

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho \left| \mathbf{u} \right|^2 \right) - \gamma p_c, \tag{6}$$

where $\mathbf{u} = (u, v)^T$ is the velocity vector, γ is the ratio of specific heats and p_c is a constant. For air, $\gamma = 1.4$ and $p_c = 0$, in which case equation 6 collapses to the ideal gas equation of state. For water, I use $\gamma = 7$ and $p_c = 3 \times 10^8$Pa.

The viscous forces are included by the source term

$$\mathbf{F}_\mu = \left[\begin{array}{ccc} 0, & \nabla \cdot \tau_\mu, & \nabla \cdot (\mathbf{u}\tau_\mu) \end{array} \right]^T. \tag{7}$$

In two-dimensional polar coordinates, the viscous stress tensor τ_μ is

$$\tau_\mu = \begin{bmatrix} \tau_{rr} & \tau_{r\theta} \\ \\ \tau_{r\theta} & \tau_{\theta\theta} \end{bmatrix}, \tag{8}$$

and the components of τ_μ are defined

$$\tau_{rr} = \mu \left(2\frac{\partial u}{\partial r} - \frac{2}{3}\nabla \cdot \mathbf{u} \right), \tag{9a}$$

$$\tau_{r\theta} = \mu \left(r\frac{\partial}{\partial r}\left(\frac{v}{r} \right) + \frac{1}{r}\frac{\partial u}{\partial \theta} \right), \tag{9b}$$

$$\tau_{\theta\theta} = \mu \left(2\left(\frac{1}{r}\frac{\partial v}{\partial \theta} + \frac{u}{r} \right) - \frac{2}{3}\nabla \cdot \mathbf{u} \right), \tag{9c}$$

where μ is the dynamic viscosity. Note that

$$\nabla \cdot \tau_\mu = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \begin{bmatrix} \tau_{rr} \\ \\ \tau r\theta \end{bmatrix} \right) + \frac{1}{r\sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta \begin{bmatrix} \tau_{r\theta} \\ \\ \tau_{\theta\theta} \end{bmatrix} \right), \tag{10}$$

due to the polar coordinate system, and

$$\mathbf{u}\tau_\mu = \left(u\tau_{rr} + v\tau_{r\theta},\, u\tau_{r\theta} + v\tau_{\theta\theta}\right)^T. \tag{11}$$

The numerical scheme used in this paper is the same as that described in King et al. (2015), but with a modification to the ghost fluid method used to model the interface, and with the viscous terms included using an operator splitting procedure. Full details of the scheme are in King et al. (2015), and are summarised in the next paragraph.

The numerical scheme is constructed from a single-phase Euler solver in conjunction with a ghost fluid method (GFM) (Fedkiw et al., 1999; Wang et al., 2006) to model the interface, which is tracked with a level set (Russo and Smereka, 2000). The single-phase Euler solver is a dimensionally split Godunov-type scheme (Godunov, 1959; Einfeldt, 1988; Ivings et al., 1998), which is first-order accurate in space and time. Numerical fluxes ($\mathbf{F}$ and $\mathbf{G}$ in equation 1) are constructed using the HLLC flux (Toro et al., 1994) and are calculated using an approximate Roe-type Riemann solver due to Hu et al. (2009). A first-order operator splitting approach is used for dimensional splitting, and also to include the geometric, gravitational and viscous source terms $\mathbf{S_r}$, $\mathbf{S}_\theta$, $\mathbf{D}$ and $\mathbf{F}_\mu$ in equation 1. $\mathbf{F}_\mu$ is calculated using a second-order central scheme within Ω and a first-order one-sided scheme on Γ. For viscous fluids the no-slip condition constrains velocities tangential to the air-water interface to be continuous across the interface. In order to enforce this, the ghost fluid method used in King et al. (2015) is modified according to Fedkiw and Liu (2001); Terashima and Tryggvason (2009).

The viscosity of both air and water depend on temperature. For air, the viscosity can be calculated using the empirical-theoretical relation of Sutherland (1893), which is given

by

$$\mu_{\text{air}} = \mu_0 \frac{T_0 + C}{T + C} \left(\frac{T}{T_0} \right)^{\frac{3}{2}}, \tag{12}$$

where $\mu_0 = 1.716 \times 10^{-5}\text{kgm}^{-1}\text{s}^{-1}$, $T_0 = 273.15\text{K}$ and $C = 110.4\text{K}$. For the International Standard Atmosphere conditions at sea level, with $T = 288.15\text{K}$, Sutherland's equation gives $\mu_{air} = 1.789 \times 10^{-5}\text{kgm}^{-1}\text{s}^{-1}$. Within the bubble the temperature is calculated according to the ideal gas equation of state $T = p/\rho R_g$, where $R_g = 287\text{Jkg}^{-1}\text{K}^{-1}$ is the gas constant for air. Sutherland's equation is known to be accurate to within 10% for temperatures below 555 Kelvin and pressures below $3.45 \times 10^6\text{Pa}$. Whilst the temperature of the air in an air gun bubble remains within this range, the pressure does not. In the results shown in King et al. (2015), for approximately the first 1ms of oscillation, the pressure in the bubble is greater than $3.45 \times 10^6\text{Pa}$. Subsequently, including when the bubble collapses, the pressure remains below this limit. During the very early stages of expansion, the motion is very close to spherically symmetric, and the bubble rise velocity is still small. The first few milliseconds of expansion are those for which which viscosity has least influence on the bubble motion. Hence, it is reasonable to use the Sutherland equation even though the pressures are not entirely within the limits for which the relation is known to be valid. In the water the temperature is close to constant, and I use $\mu_{\text{water}} = 10^{-3}\text{kgm}^{-1}\text{s}^{-1}$ (Langhammer and Landrø, 1993).

Polar coordinate systems contain singularities both at the origin and along the polar axes. It is known (Kane et al., 2000; Kifonidis et al., 2003, 2006) that numerical instabilities may occur along the polar axes in polar coordinate systems, the main symptom of which is faster growth of Rayleigh-Taylor instabilities, and in some cases axial jets. This instability affects the present scheme. As the bubble collapses, a physical jet forms on the underside and begins to pierce upwards through the bubble. Due to the numerical instability an

opposing (non-physical) jet forms, and the collision of the two jets causes the simulation to break down shortly after one full oscillation. This problem was noted by King et al. (2015) who suggested that a scheme using an irregular grid and a cylindrical coordinate system may solve this problem.

In the results presented in the following section, "viscous model" refers to the model described here, whilst "inviscid model" refers to the model of King et al. (2015).

Model parameters and initial conditions

The initial conditions are given by

$$
\begin{aligned}
&(\rho,\, \mathbf{u},\, p,\, \gamma,\, p_c) \\
&= \begin{cases} \big(102,\, 0,\, 8.85 \times 10^6 + p_h,\, 1.4,\, 0\big) & \text{if } 0 \leq r \leq 0.1, \\ \big(\rho_w,\, 0,\, p_w,\, 7.0,\, 3 \times 10^8\big) & \text{if } 0.1 < r \leq R_D, \end{cases}
\end{aligned} \tag{13}
$$

where $\rho_w = 1000$, $p_w = p_{atm} + \rho_w g d + p_h$, $p_h = -\rho g r \cos\theta$, $p_{atm} = 1.01325 \times 10^5$ and d is the depth of the gun. All quantities are in S.I. Units, such that radius, density, velocity and pressure have units of m, kgm^{-3}, ms^{-1} and Pa, respectively. p_c has units of Pa, and γ is dimensionless. In all cases, $d = 7.7\text{m}$, $g = 9.81\text{ms}^{-2}$, the CFL number (Courant et al., 1928, 1967) is 0.8 and $\delta\theta = \pi/50$ radians. Except where explicitly stated otherwise, $\delta r = 0.02\text{m}$ and $R_D = 1\text{m}$.

This set of initial conditions represents a 250 cubic inch (4.01 litres) air gun, fired at a pressure of 2000 psi (13.79 MPa), at a depth of 7.7metres. Note that the discrepancy in initial pressures - $8.85 \times 10^6\text{Pa} \approx 1300\text{psi}$ - is intentional, and is designed to account for the process by which air is released from the gun. Note here that the purpose of this paper is to model the effects of viscosity in a more realistic way than has been acheived in previous

models. It is not the aim of this paper to achieve a good match with measured data. If this were the aim, a more traditional model such as, for example, that of Ziolkowski and Metselaar (1984) would perform better.

NUMERICAL RESULTS

The effects of viscosity on bubble signature

The interface position R_{int} is defined as the radius of a sphere with the same volume as the bubble. The interface pressure P_{int} is the average pressure over the bubble surface.

Figure 2 shows the variation of R_{int} and P_{int} with time for the inviscid and viscous models, with $R_D = 1$m and $\delta r = 0.02$m. The traces in Figure 2 match very closely, to within 0.5% prior to $t = 0.07$ seconds. During the final few milliseconds of collapse, the differences between the inviscid and viscous models increase. The viscous model shows marginally less damping than the inviscid model, with a higher peak pressure as the bubble collapses. This is in partial agreement with the results of Langhammer and Landrø (1993), who observed a reduction in damping on collapse, and a reduction in the bubble period with increasing viscosity. The modelled results show no significant change in bubble period.

The viscous model is run on a domain of radius $R_D = 1.2$m, with $\delta r = 0.02$m. The slightly larger domain prevents the following results from being affected by errors due to lower order discretisation of $\mathbf{F}_\mu$ on the domain boundary Γ. The ratio of the magnitude of the viscous terms to the magnitude of the rate of change of conservative properties in equation 1 - density, radial momentum, polar momentum and energy - is

$$\Phi_\mu = \frac{|\mathbf{F}_\mu|}{|\partial \mathbf{U}/\partial t|} \tag{14}$$

and is calculated at every point in the domain. This quantity gives a measure of the relative importance of viscous terms. The density component of this quantity is zero, as the density component of $\mathbf{F}_\mu$ is zero by definition.

Figure 3a shows the spatial variation of Φ_μ along a line from the origin to $(1, \pi/2)$ at $t = 56.1$ms. At this point in time the air-water interface is at approximately $r = 0.4$m. Within the bubble ($r < 0.4$m), there is significant variation in Φ_μ, with values ranging between 3.5×10^{-5} and 0.4 for momentum, and 10^{-8} and 10^{-5} for energy. At the air-water interface there is a decrease of at least two orders of magnitude in Φ_μ for momentum. Outside the bubble ($r > 0.4$m), Φ_μ decreases with increasing radius, with small scale spatial variation of approximately one order of magnitude.

Figure 3b shows the variation of Φ_μ with time, at the point $(1, \pi/2)$. The main peaks in the trace for radial momentum, at $t = 0.009$ and $t = 0.06$ seconds, correspond to the times at which the radial velocity at this point is at a maximum during expansion and collapse. The minima in the traces for radial momentum and energy at approximately $t = 0.033$ seconds correspond to the radial motion being zero as the bubble is at its maximum volume. The variation in all three traces with period of between 0.001 and 0.005 seconds is due to the spatial discretisation of the interface. The motion of the interface is close to, but not exactly, radial. The interface expands slightly faster for $\theta < \pi/2$ than for $\theta > \pi/2$, and this occasionally causes small steps in the interface, as it expands into a new cell at θ earlier than at $\theta + \delta\theta$. These steps are a source of vorticity which propagates outwards, and cause the variation with a period between 0.001 and 0.005 seconds in Figure 3b. This is a consequence of the discretisation and the numerical scheme, and is reduced with mesh refinement.

Most importantly, Figure 3b shows that the relative contribution of viscous terms is

always less than 10^{-4} at $r = 1$ metre, and for most of the simulation is at least two orders of magnitude smaller. Hence, the assumption of inviscid flow on which the boundary condition is based is reasonable at this radius. If the contribution of viscous terms were significant on R_D, R_D could simply be increased until this were not the case.

Table 1 shows the effects of changing the viscosity on the peak pressure developed in the bubble during collapse. Note that the change of the ghost fluid method (GFM) from inviscid to viscous significantly increases the collapse pressure. The effect of increasing the values of viscosity is smaller, although an increase in μ_{water} also causes an increase in the collapse pressure, as observed by Langhammer and Landrø (1993). The weak influence of viscosity on bubble radius and bubble pressure is in agreement with the arguments of Taylor and Davies (1943) and Ziolkowski (1970). It is worth noting that there is numerical viscosity inherent in the scheme, and this may also play a role. Some numerical viscosity is necessary for the stability of the scheme. Discussions of the dissipation mechanisms of Godunov-type schemes is given by Park and Kwon (2003) and Xu and Li (2001). Xu and Li (2001) point out that the effective viscosity of Godunov-type schemes depends on the flow solution and the mesh construction, and is not necessarily consistent with the Navier-Stokes viscous terms. That the numerical viscosity is inherent in the scheme means that it cannot be varied independently of the mesh or the building blocks of the scheme itself, such as the reconstruction procedures or fluxes. Hence, no attempt has been made to determine the effect of the inherent numerical viscosity. Given the dependence of the numerical viscosity on the mesh and the scheme, the complexity of the flow around the bubble and the long run times ($\mathcal{O}\left(10^5\right)$ time-steps) associated with bubble oscillations, it may not be possible to quantify the effect of the inherent numerical viscosity on bubble oscillations. The numerical viscosty is not directly influenced by the values given to μ, or the choice of GFM, and it

may be assumed that the effect of the physical viscosity is independent of the effect of the numerical viscosity. Therefore the conclusion from Table 1, that the effect of the value given to μ is much smaller than the effect of the application of the no-slip condition, is valid.

The effects of viscosity on bubble structure

I run both the viscous and inviscid models with $R_D = 1$m and $\delta r = 0.02$m. In the viscous model, the values of viscosity set out in the section Numerical Method, above, are used. Figure 4 shows contours of velocity magnitude at $t = 33.6$ms (approximately the time at which the maximum value of R_{int} occurs), for (a) the inviscid and (b) the viscous model. In both models there is a change in the velocity across the interface. Whilst in the inviscid model the velocity is discontinuous here, in the viscous model there is no discontinuity - the velocity changes smoothly across the interface. There are boundary layers both inside and outside the bubble. In the inviscid model the maximum velocity occurs at the interface, whilst in the viscous model the maximum velocity occurs about 0.1m from the interface, inside the bubble. It is the modification of the GFM to enforce the no-slip condition which leads to these boundary layers. Note the kink in the contours on the interface at $(x, z) = (0.4, 0.2)$ in Figure 4a. This is due to the discretisation of the interface as discussed earlier.

Figure 5 shows the radial and polar components of velocity along the line between the origin and $(R_D, \pi/2)$ just before the bubble reaches its maximum volume, at $t = 32.3$ms. The solid line corresponds to the inviscid model, and the dashed line to the viscous model. The location of the interface is approximately $r = 0.45$. Note that in both models the radial velocity component is continuous across the interface, although with discontinuous

gradient, and there is little difference in radial velocity component between models. For the inviscid model, the polar component of velocity is discontinuous. In the viscous model, the polar velocity is continuous across the interface. There is still a large change in the polar component of velocity across the interface, but it varies smoothly. Far from the bubble, the velocity fields are very similar for the inviscid and viscous models, which reinforces the justifications of Taylor and Davies (1943) and Ziolkowski (1970) and the results of Langhammer and Landrø (1993) that water viscosity has little effect on the signals emitted by the bubble. Within the bubble there are significant differences between the viscous and inviscid models, as the no-slip condition causes a boundary layer within the bubble. The effects of this small change accumulate over time leading to differences in velocity fields throughout the bubble. Given the low density of the bubble contents, it is likely that these structural changes in the bubble have little effect on the signals emitted by the bubble.

Figure 6 shows contours of the magnitude of polar velocity for both (a) inviscid and (b) viscous models with $R_D = 1\text{m}$ and $\delta r = 0.01\text{m}$, at $t = 32.5\text{ms}$ - again at approximately the moment of maximum expansion. At this point in time the bubble is still close to spherical, and has risen only a small distance. Therefore the polar velocity component is a good proxy for the transverse velocity at the interface. The boundary layers both sides of the interface are clearly visible in Figure 6b. Within the bubble the boundary layer thickness is approximately 0.06m, and outside the bubble is 0.03m. The presence of the boundary layers is due to the no-slip condition. The thickness of the boundary layers is influenced by, amongst other things, the magnitude of the viscosity.

Based on the polar velocity component at the interface, the maximum polar velocity in the bubble, and the fact that far from the bubble the polar velocity falls to zero, I estimate free-stream velocities for the boundary layers to be $V_{fs,air} = 0.4\text{ms}^{-1}$ and $V_{fs,water} =$

0.2ms^{-1}. At the moment of maximum expansion the density of the bubble is 1.1kgm^{-3}, and the density of the water is 1000kgm^{-3}. Taking $\mu_{air} = 4 \times 10^{-6}\text{kgm}^{-1}\text{s}^{-1}$ (based on Sutherland (1893)) and $\mu_{water} = 10^{-3}\text{kgm}^{-1}\text{s}^{-1}$, and assuming the same characteristic length scale both inside and outside of the bubble, I find that the Reynolds number for the motion transverse to the interface is 1.8 times greater in the water than in the air, at this instant in time. This difference in Reynolds numbers shows that the relative importance of viscous forces in the water is less than in the air. This contributes to the small influence of viscosity on the motion away from the bubble compared with the influence of viscosity within the bubble. Furthermore, boundary layers are thicker in regions with lower Reynolds numbers, as observed in Figure 6b.

Figure 7 shows the shape of the bubble at various stages of collapse, with $R_D = 1\text{m}$ and $\delta r = 0.02\text{m}$, for the viscous (dashed line) and inviscid (solid line) models. The shapes closely match during the early stages of collapse (frames (a) to (c)), but begin to differ towards the end of collapse (frames (d) to (f)). This is unsurprising. At this stage in the oscillation, the surface of the bubble is unstable, and the bubble shape at the end of collapse is highly sensitive to small changes in shape during expansion and the early stages of collapse. This sensitivity is also observed in King et al. (2015). Figure 7 also shows a reduction in rise rate when viscosity is included. In all frames, the line representing the surface of the bubble in the viscous model lies slightly lower than that for the inviscid model. By numerically integrating both the density and the first moment of density over the volume of the bubble the vertical location of the centre of mass of the bubble z_b is calculated. Table 2 shows the distance risen by the bubble at $t = 68\text{ms}$, with both the viscous and inviscid versions of the GFM, with and without the viscous terms. The influence of the viscous terms on the distance risen is small, at 2.2%, whilst the influence of the viscous GFM on the distance

risen is significant, at approximately 20%.

The drag on a body moving through a fluid is composed to two parts - form drag and skin friction drag. Form drag is due to the size and shape of the body, and the energy imparted on the fluid by the body as the fluid moves around the body. Skin friction drag is due to the friction of the fluid against the surface of the body, which causes energy to be transferred from the body to the fluid, and is influenced by viscosity and the surface area of the body. For a bluff body such as a sphere, form drag dominates. However, the influence of skin friction drag is not negligible. The inviscid model is unable to capture the skin friction drag on the bubble as it rises, as the no-slip condition is not applied at the interface, and the forces due to viscosity are not included. The viscous model captures the boundary layers on the bubble surface. The decrease in bubble rise rate in the viscous model is due to skin friction drag.

CONCLUSIONS

I present an air gun bubble model in which the two-dimensional compressible Navier-Stokes equations are solved numerically to simulate a cylindrically symmetric air gun bubble. This approach allows more flexibility regarding the assumptions and approximations made in air gun modelling than in classical air gun bubble models. The model presented in this paper includes viscosity in a way that is more realistic than previous models.

When viscosity is included in the model, the resulting bubble oscillations show slightly reduced damping. This is in agreement previous experimental observations, and in disagreement with previous models. Changing the values of viscosity in the model is found to have only a small effect on signatures, whilst the no-slip condition at the air water interface

has significant influence. There is unavoidable numerical viscosity inherent in the scheme, necessary for stability, which is not necessarily of the same form as the viscous terms in the governing equations. However, this is not directly influenced by the viscosity, and for the purposes of this investigation may be ignored.

The application of the no-slip condition at the interface in the viscous model causes boundary layers within and around the bubble, and the velocity field is more realistic. The no-slip condition allows the model to capture skin friction drag on the bubble as it rises, reducing the bubble rise rate. The modelled results show a reduction in the distance risen by the bubble during the first oscillation of about 20% in the viscous model compared with the inviscid model. This difference is predominantly due to the no-slip condition.

The observation of a reduction in damping and in bubble rise rate may not be independent, as a deeper bubble undergoes a stronger collapse due to the higher hydrostatic pressure in the water. Whilst the effect of viscosity on the bubble signature is weak, the effect on the rise rate is significant. In marine seismic exploration, it is desirable to know the location of the source as precisely as possible. Correctly calculating the rise rate of the bubble is therefore important. The results presented here show that viscosity - and in particular the no-slip condition at the interface - influences bubble rise rates. Viscosity should be included in bubble rise rate calculations in future models.

ACKNOWLEDGMENTS

I thank Petroleum Geo-Services (PGS) for funding this research. I thank Anton Ziolkowski for many useful discussions, and for helpful comments on the manuscript.

REFERENCES

Blake, J. R., G. S. Keen, R. P. Tong, and M. Wilson, 1999, Acoustic cavitation: the fluid dynamics of non-spherical bubbles: Philosophical Transactions of the Royal Society of London A, **357**, 251–267.

Bornhorst, W. J., and G. N. Hatsopoulos, 1967, Bubble-growth calculation without neglect of interfacial discontinuities: Journal of Applied Mechanics, **34**, 847–853.

Courant, R., K. Friedrichs, and H. Lewy, 1928, ber die partiellen differenzengleichungen der mathematischen physik: Mathematische Annalen, **100**, 32–74.

———, 1967, On the partial difference equations of mathematical physics: IBM Journal of Research and Development, **11**, 215–234.

Cox, E., A. Pearson, J. R. Blake, and S. R. Otto, 2004, Comparison of methods for modelling the behaviour of bubbles produced by marine seismic airguns: Geophysical Prospecting, **52**, 451–477.

Dragoset, W. H., 1984, A comprehensive method for evaluating the design of air guns and air gun arrays: The Leading Edge, **3**, 52–61.

Einfeldt, B., 1988, On Godunov-type methods for gas dynamics: SIAM Journal of Numerical Analysis, **25**, 294–318.

Fedkiw, R. P., T. Aslam, B. Merriman, and S. Osher, 1999, A non-oscillatory Eulerian approach to interfaces in multimaterial flows - the ghost fluid method: Journal of Computational Physics, **152**, 457–492.

Fedkiw, R. P., and X.-D. Liu, 2001, The ghost fluid method for viscous flows, *in* Innovative methods for numerical solution of partial differential equations: World Scientific.

Gilmore, F. R., 1952, Collapse of a spherical bubble: Technical Report No. 26-4, Hydrodyn. Lab., Calif. Inst. Tech.

Godunov, S. K., 1959, A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics: Matematicheskii Sbornik, **47(89)**, 271–306.

Hu, X. Y., N. A. Adams, and G. Iaccarino, 2009, On the HLLC Riemann solver for interface interaction in compressible multi-fluid flow: Journal of Computational Physics, **228**, 6572–6589.

Ivings, M. J., D. M. Causon, and E. F. Toro, 1998, On Riemann solvers for compressible liquids: International Journal for Numerical Methods in Fluids, **28**, 395–418.

Kane, J., D. Arnett, B. A. Remington, S. G. Glendinning, G. Bazan, E. Muller, B. A. Fryxell, and R. Teyssier, 2000, Two-dimensional versus three-dimensional supernova hydrodynamic instability growth: The Astrophysical Journal, **528**, 989–994.

Kifonidis, K., T. Plewa, H. T. Janka, and E. Muller, 2003, Non-spherical core collapse supernovae I. Neutrino-driven convection, Rayleigh-Taylor instabilities, and the formation and propagation of metal clumps: Astronomy and Astrophysics, **408**, 621–649.

Kifonidis, K., T. Plewa, L. Scheck, H.-T. Janka, and E. Muller, 2006, Non-spherical core collapse supernovae II. The late-time evolution of globally anistropic neutrino-driven explosions and their implications for SN 1987 A: Astronomy and Astrophysics, **453**, 661–678.

King, J. R. C., A. M. Ziolkowski, and M. Ruffert, 2015, Artificial boundary conditions for simulations of oscillating bubbles using the non-linear acoustic approximation: Journal of Computational Physics, **284**, 273–290.

Kirkwood, J. G., and H. A. Bethe, 1942, Progress report on 'The pressure wave produced by an underwater explosion I': Technical Report No. 588, Office of Scientific Research and Development, US Navy.

Lamb, H., 1923, The early stages of submarine explosion: Philosophical Magazine, **45**,

257–265.

Landrø, M., 1992, Modelling of GI gun signatures: Geophysical Prospecting, **40**, 721–747.

Landrø, M., and R. Sollie, 1992, Source signature determination by inversion: Geophysics, **57**, 1633–1640.

Langhammer, J., and M. Landrø, 1993, Experimental study of viscosity effects on air-gun signatures: Geophysics, **58**, 1801–1808.

Laws, R. M., L. Hatton, and M. Haartsen, 1990, Computer modelling of clustered airguns: First Break, **8**, 331–338.

Lezzi, A., and A. Prosperetti, 1987, Bubble dynamics in a compressible liquid. Part 2. Second-order theory: Journal of Fluid Mechanics, **185**, 289–321.

Liu, T. G., B. C. Khoo, and C. W. Wang, 2005, The ghost fluid method for compressible gas-water simulation: Journal of Computational Physics, **204**, 193–221.

Park, S. H., and J. H. Kwon, 2003, On the dissipation mechanisms of Godunov-type schemes: Journal of Computational Physics, **188**, 524–542.

Plesset, M. S., 1949, The dynamics of caviation bubbles: Journal of Applied Mechanics, **16**, 277–282.

Prosperetti, A., and A. Lezzi, 1986, Bubble dynamics in a compressible liquid. Part 1. First-order theory: Journal of Fluid Mechanics, **168**, 457–478.

Rayleigh, J. W. S., 1917, On the pressure developed in a liquid during the collapse of a spherical cavity: Philosophical Magazine, **34**, 94–99.

Russo, G., and P. Smereka, 2000, A remark on computing distance functions: Journal of Computational Physics, **163**, 51–67.

Sutherland, W., 1893, The viscosity of gases and molecular force: Philosophical Magazine Series 5, **36**, 507–531.

Taylor, G. I., and R. M. Davies, 1943, *in* The motion and shape of the hollow produced by an explosion in a liquid: Office of Naval Research, Dept. of the Navy.

Terashima, H., and G. Tryggvason, 2009, A front-tracking/ghost fluid method for fluid interfaces in compressible flows: Journal of Computational Physics, **228**, 4012–4037.

Toro, E. F., M. Spruce, and M. Speares, 1994, Restoration of the contact surface in the HLL-Riemann solver: Shock Waves, **4**, 25–34.

Wang, W., T. G. Liu, and B. C. Khoo, 2006, A real ghost fluid method for the simulation of multimedium compressible flow: SIAM Journal of Scientific Computing, **28**, 278–302.

Xu, K., and Z. Li, 2001, Dissipative mechanisms in Godunov-type schemes: International Journal for Numerical Methods in Fluids, **37**, 1–22.

Ziolkowski, A., 1970, A method for calculating the output pressure waveform from an air gun: Geophysical Journal of the Royal Astronomical Society, **21**, 137–161.

———, 1998, Measurement of air-gun bubble oscillations: Geophysics, **63**, 2009–2024.

Ziolkowski, A. M., and G. Metselaar, 1984, The pressure wavefield of an air gun array: Expanded Abstracts 54th SEG Meeting, Atlanta, 274–276.

LIST OF TABLES

LIST OF FIGURES

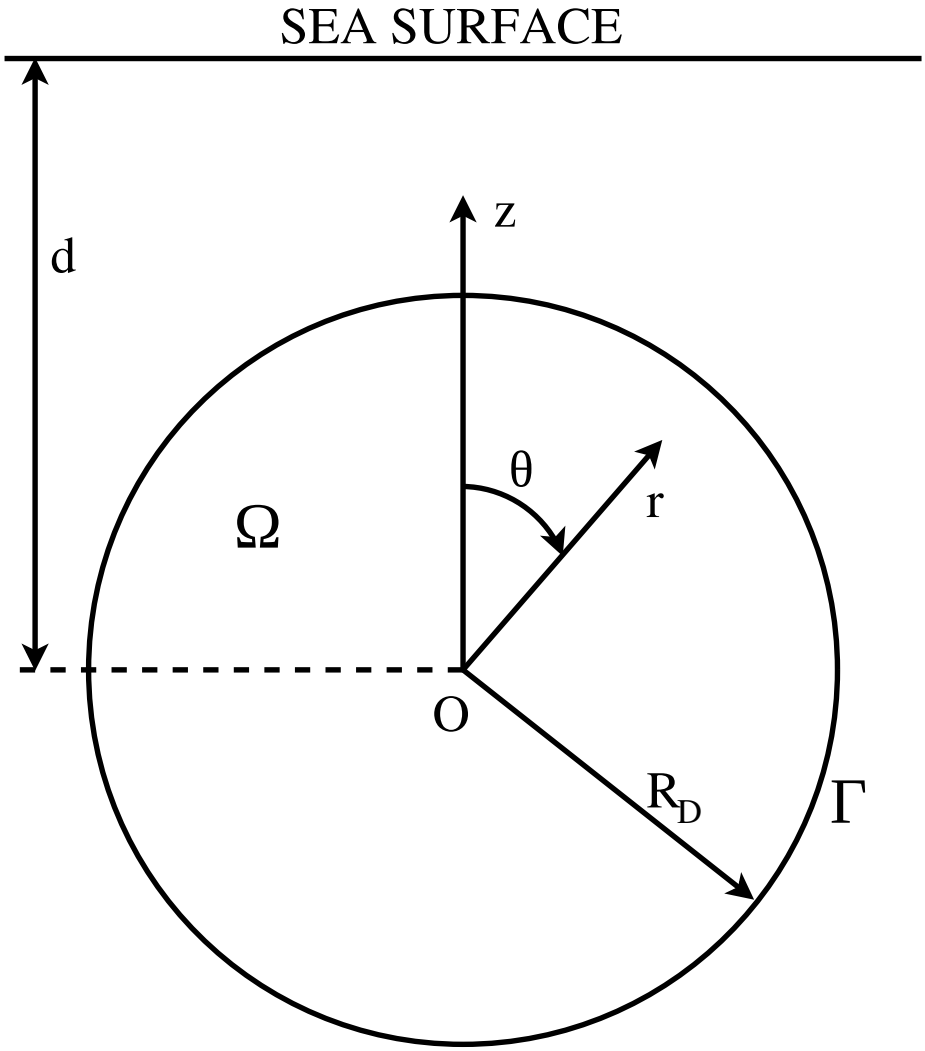
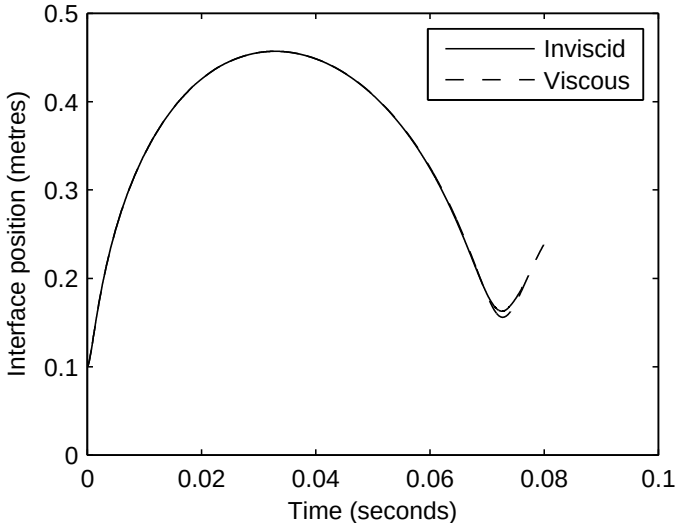
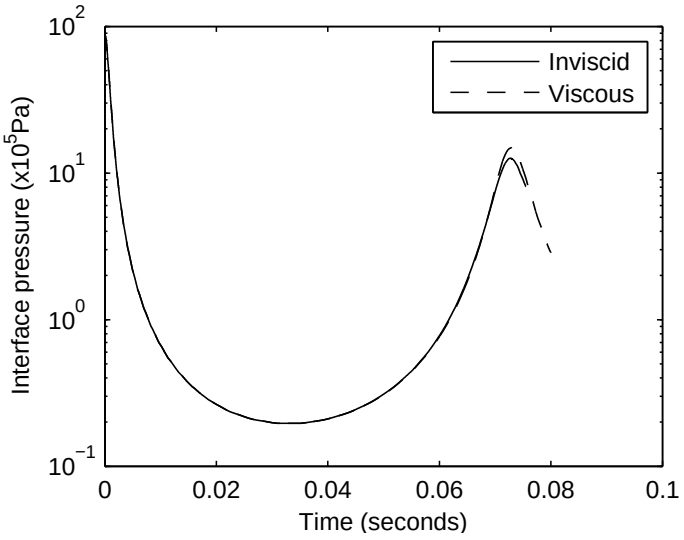


Figure 1: A schematic diagram of the computational domain.

–



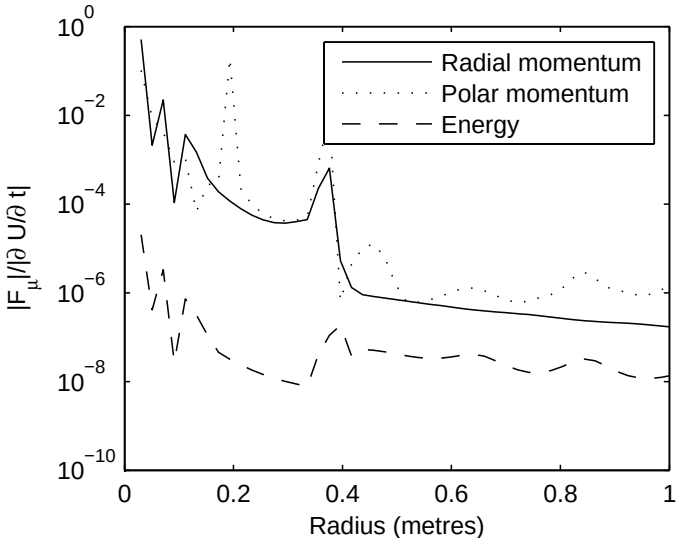
(a)



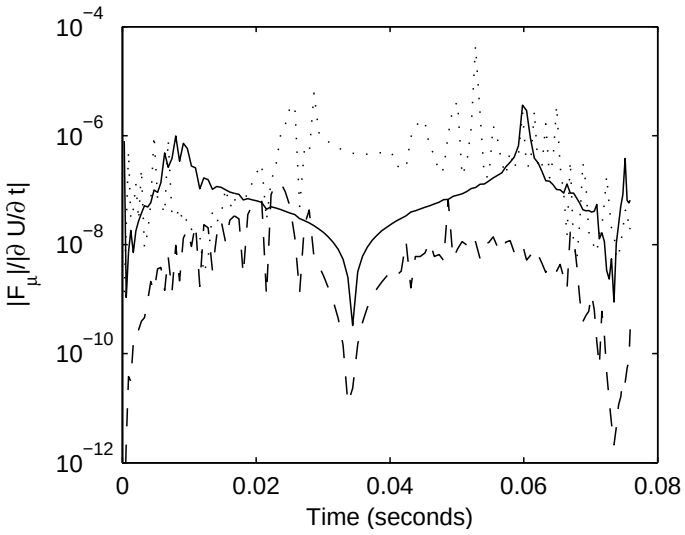
(b)

Figure 2: Variation of interface position (a) and interface pressure (b) with time. Dashed line - viscous model; solid line - inviscid model.

–



(a)



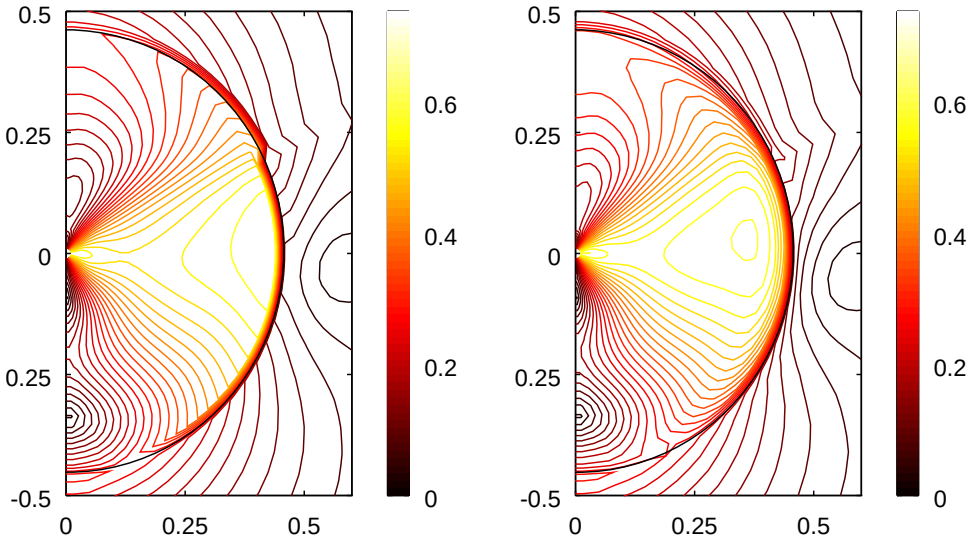
(b)

Figure 3: The ratio of the magnitude of viscous terms to the magnitude of time derivatives. (a) - Variation along the line $\theta = \pi/2$ at $t = 56.1$ms and (b) variation with time at the point $(1, \pi/2)$. Solid line - radial momentum; dotted line - polar momentum; dashed line - energy.

–

GFM	μ_{air}	μ_{water}	P_{collapse} ($\times 10^6$Pa)
Inviscid	0	0	1.261
Inviscid	Sutherland	10^{-3}	1.283
Viscous	0	0	1.485
Viscous	Sutherland	10^{-3}	1.487
Viscous	Sutherland	0.489	1.529
Viscous	0	10^{-3}	1.486
Viscous	10^{-5}	10^{-3}	1.492

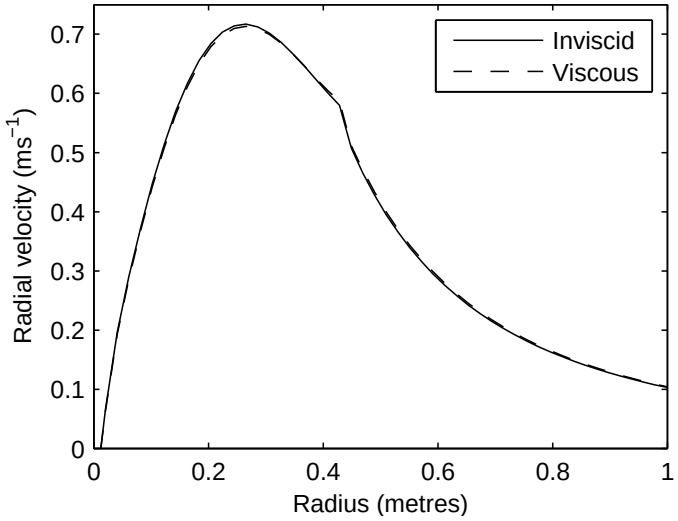
Table 1: The peak pressure in the bubble during collapse, for varying values of viscosity, and different forms of the ghost fluid method (GFM). 'Sutherland' denotes the viscosity obtained using equation 12. Values of viscosity have units $\text{kgm}^{-1}\text{s}^{-1}$.



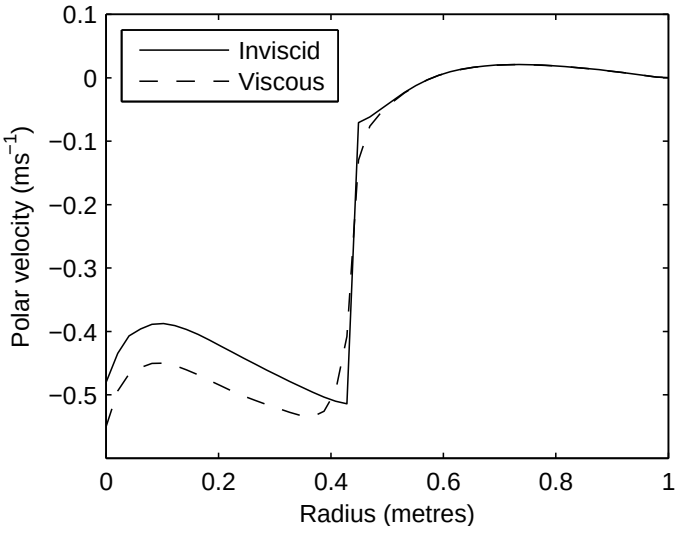
(a) (b)

Figure 4: Contours of velocity magnitude at $t = 33.6$ms for inviscid (a) and viscous (b) models. The abscissa and the ordinate represent the x- and z-coordinates respectively, in metres. The solid black line is the air-water interface.

–



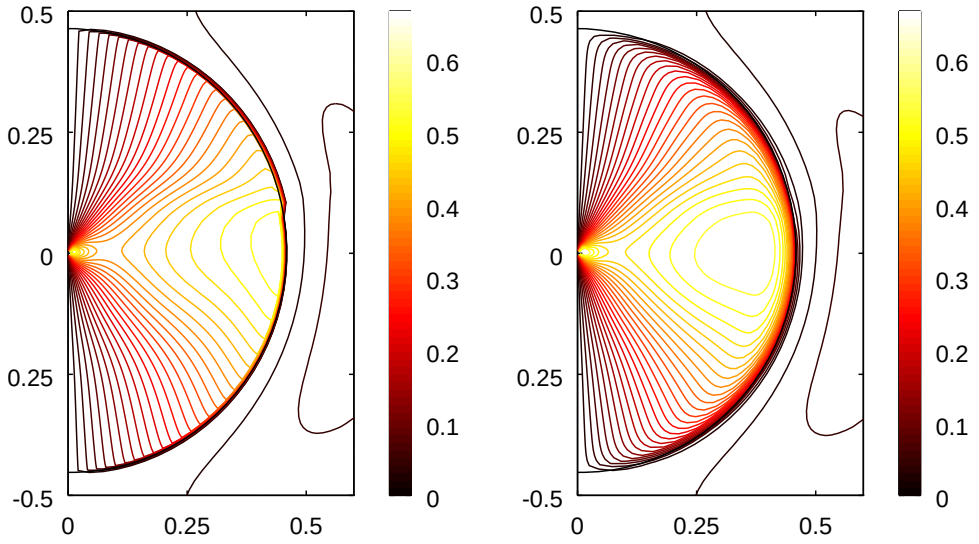
(a)



(b)

Figure 5: Variation of radial and polar velocity components with radius along $\theta = \pi/2$ at $t = 32.2$ms, for inviscid (solid line) and viscous (dashed line) models. The interface is located at approximately $r = 0.45$m.

–



(a) (b)

Figure 6: Contours of the magnitude of the polar component of velocity at $t = 32.5$ms for inviscid (left) and viscous (right) models, with $\delta r = 0.01$. The abscissa and the ordinate represent the x- and z-coordinates respectively, in metres. The solid black line is the air-water interface.

–

GFM	μ_{air}	μ_{water}	z_b (m)	% change
Inviscid	0	0	0.0270	0
Inviscid	Sutherland	10^{-3}	0.0264	-2.2%
Viscous	0	0	0.0215	-20.4%
Viscous	Sutherland	10^{-3}	0.0221	-18.1%

Table 2: The distance risen by the bubble at $t = 68$ms, for varying values of viscosity, and different forms of the ghost fluid method (GFM). 'Sutherland' denotes the viscosity obtained using equation 12. Values of viscosity have units $\text{kgm}^{-1}\text{s}^{-1}$. The final column shows the percentage change in distance risen compared with the inviscid model in the first row.

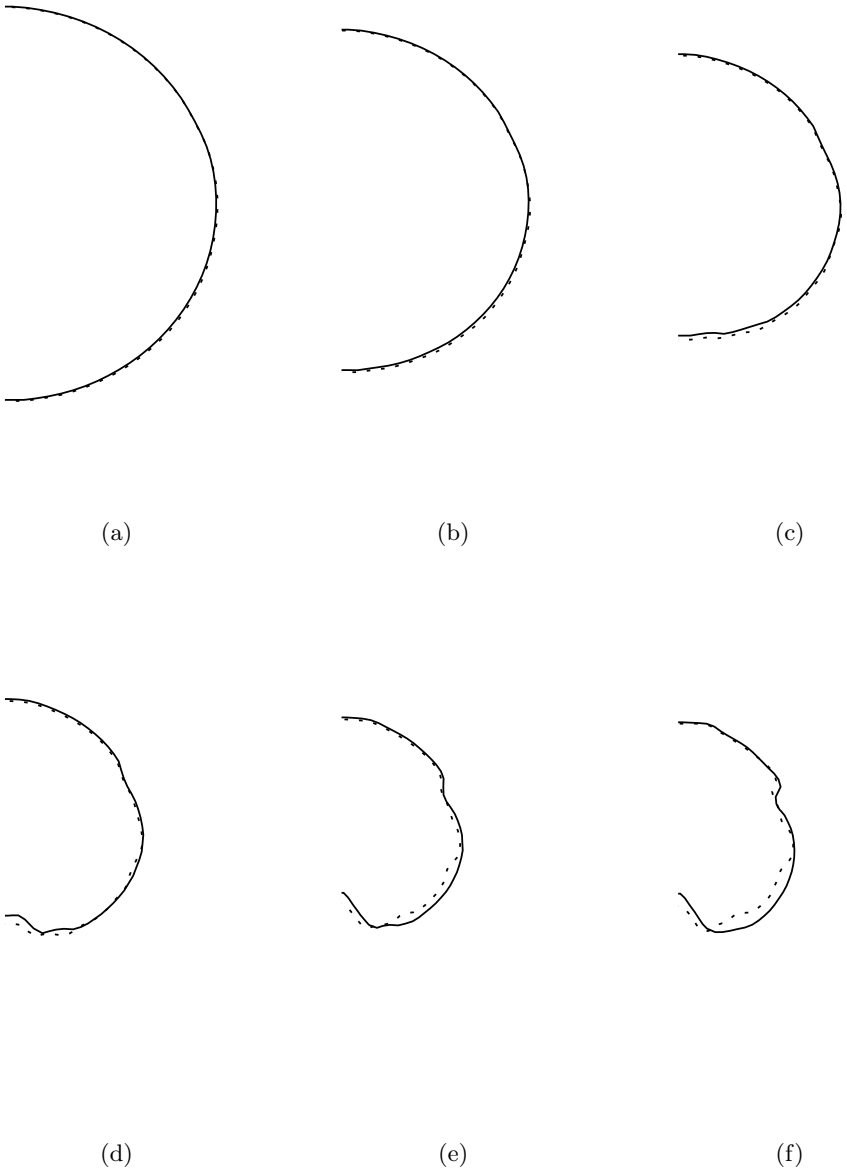


Figure 7: The shape of the bubble at different times during collapse, for viscous (dashed line) and inviscid (solid line) models. (a) $t = 62.7$ms; (b) $t = 65.3$ms; (c) $t = 67.7$ms; (d) $t = 69.9$ms; (e) $t = 71.8$ms; (f) $t = 73.7$ms.

–

Bubble-ghost interactions

Jack King*

* *The Grant Institute,*

School of GeoSciences,

University of Edinburgh,

James Hutton Road,

Edinburgh,

EH9 3FE,

UK

(February 23, 2015)

Running head: **Bubble-ghost interactions**

ABSTRACT

Sea surface reflections may be included in air-gun bubble models by creating a virtual image of the gun (termed the "ghost") which is a reflection of the gun in the sea surface, usually regarded as plane. In current air-gun bubble modelling, the interaction between the bubble and the ghost is treated with spherical symmetry. I present a two-dimensional axisymmetric finite-volume model of an air-gun bubble, in which the effects of the sea surface are included through the use of a novel artificial boundary condition, which passes the ghost wave into the computational domain. The interaction between the bubble and the ghost wave is then treated axisymmetrically. The ghost is found to reduce the damping and the period of the bubble. At the bubble surface the ghost wave is both transmitted into the bubble and

reflected back into the water. The bubble focusses the ghost wave on (or in reality near) the bubble centre where it is reflected. This process repeats for a few oscillations with diminishing magnitude, as the energy of the ghost in the bubble is transmitted into the water through a series of pulses. This gives rise to oscillations at frequencies between 400 and 600Hz in near-field measurements shortly after the impact of the ghost wave on the bubble. This theory of bubble-ghost interactions is supported by data.

INTRODUCTION

In marine seismic exploration, in order to achieve accurate deconvolution of the seismic data, it is necessary to have an accurate knowledge of the signal produced by the air gun source or air-gun array. Industry uses modelling to determine this signal, and also in the design of arrays. Modelling requires an understanding of the influence of the sea surface on the bubble in order to achieve accurate results.

Air-gun bubbles were first modelled by Ziolkowski (1970), who used a scheme developed by Gilmore (1952), following from a line of work on cavitation and underwater explosions leading back through Kirkwood and Bethe (1942) and Lamb (1923) to Rayleigh (1917). This same line of work led to the widely used Rayleigh-Plesset equation (Plesset, 1949; Prosperetti and Lezzi, 1986; Lezzi and Prosperetti, 1987). Air-gun models were further developed by, for instance, Dragoset (1984), Laws et al. (1990), Landrø (1992) and Landrø and Sollie (1992), based on the same approximations as Ziolkowski (1970), but with viscous terms added to provide more damped results. Ziolkowski and Metselaar (1984) presented an improvement to the model of Ziolkowski (1970) which aimed to capture the thermodynamics of the bubble more realistically. Ziolkowski (1998) presented a method to calculate the wavefield produced by an air gun from near-field measurements. The method contains an improvement to the equation of motion used in the model of Ziolkowski (1970), which removes the need for a complex transition between the near-field and far-field. All these air gun models are based on the assumptions of homogeneous spherical bubbles. Cox et al. (2004) reviewed several different air gun bubble models, including one using boundary element methods similar to Blake et al. (1999). In these models the bubbles are not constrained to be spherical, but can deform due to gravity. King et al. (2015) developed

an artificial boundary condition for use in finite volume simulations of oscillating bubbles, with a focus on air gun bubbles. King et al. (2015) demonstrated their method in one and two dimensions, showing the results of long run time axisymmetric fixed-grid finite-volume simulations of oscillating bubbles on highly truncated domains. They did not account for the sea surface in any way.

Signals from air-gun bubbles are reflected from the sea surface with a reflection coefficient of close to -1 in calm seas. In the model of Ziolkowski (1970), the sea surface is accounted for using a "ghost" bubble, which is a virtual image of the real air-gun bubble reflected in the sea surface. The near-field signal is assumed to be the superposition of the signals from the real and ghost guns. A method to calculate the far-field signatures of arrays from near-field measurements was developed by Ziolkowski et al. (1982) and Parkes et al. (1984). This method uses the superposition theory to determine the pressure field. Ziolkowski et al. (1982) and Parkes et al. (1984) explained that the effect of interactions between guns, and also the interactions with the sea surface, may be accounted for by a modulation of the hydrostatic pressure at each gun, by the combined signals of all guns and ghosts. This method has been used to account for the sea surface by, for instance, Landrø and Sollie (1992) and Ziolkowski (1998). The more recent work of Cox et al. (2004) follows Ziolkowski (1970), and does not include any interaction between the bubble and the sea surface.

I present an extension to the non-linear acoustic approximation (NLAA) artificial boundary condition of King et al. (2015), which allows the effects of the sea surface to be included in a two-dimensional finite volume simulation of an air gun bubble. In this way the ghost wave is transmitted into the computational domain and allowed to interact with the bubble. I present numerical results showing this phenomenon. The results suggest a novel mechanism for the creation of higher frequencies in air gun signatures as the ghost wave reaches

the bubble. I present experimental observations supporting my theory.

The layout of this paper is as follows. In the next section I provide a summary of the NLAA artificial boundary condition of King et al. (2015). I then present an extension to the boundary condition which accounts for the effects of the sea surface, and allows the ghost wave to be transmitted into the computational domain. There follows a brief summary of the numerical model in which my method is implemented, which is the same as the model developed and described fully in King et al. (2015). Numerical results are then presented that show the influence of the ghost on the bubble, and capture the asymmetric interaction between the bubble and the ghost. I then provide experimental evidence in support of the theory put forward in Numerical Results. Finally, I draw conclusions from the results presented.

THE NLAA ARTIFICIAL BOUNDARY CONDITION

The NLAA was developed by Ziolkowski (1998), as part of a method to calculate the wavefield produced by an air gun from near-field measurements, made close to, or even inside, the bubble. King et al. (2015) use the NLAA to derive an artificial boundary condition for use in numerical simulations of the Euler equations. In King et al. (2015) the artificial boundary condition is implemented in a two-dimensional finite volume scheme for simulations of air-gun bubbles. However, the sea surface is not included. In this section I provide a summary of the NLAA boundary condition, which contains the pertinent results that are the basis for an extension of the method to include the effects of the sea surface. This extension is presented in the next section. Full details of the NLAA boundary condition are presented in King et al. (2015).

Consider a Cartesian coordinate system, with origin located on an air gun, and z increasing with decreasing depth. The gun is assumed to be stationary in still water, and hence I make the approximation that all motion is axisymmetric about the z-axis. I define a polar coordinate system with the same origin such that the position vector $\mathbf{r} = (r, \theta)$ in the polar coordinate system is defined by $r = \left(x^2 + y^2 + z^2\right)^{\frac{1}{2}}$ and $\cos\theta = z/r$.

Let Ω be a spherical domain with radius R_D. The boundary of Ω is denoted by Γ. Ω is centred on the origin, and R_D is large enough that the bubble produced by the air gun remains completely within Ω during the period of interest. For simplicity, I neglect the physical body of the gun, the gun housing and all cables, and consider only the bubble produced when the gun is fired. Throughout this paper I consider inviscid fluids, which are governed by the Euler equations. Within Ω, I numerically solve the Euler equations. I apply the NLAA artificial boundary condition on Γ.

I write the Euler equations in two-dimensional polar coordinates as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial r} + \frac{\partial \mathbf{G}}{\partial \theta} + \mathbf{S_r} + \mathbf{S}_\theta = \mathbf{D}, \tag{1}$$

where the vector of conserved properties is

$$\mathbf{U} = \left[\begin{array}{cccc} \rho, & \rho u, & \rho v, & E \end{array} \right]^T, \tag{2}$$

the fluxes of $\mathbf{U}$ are

$$\mathbf{F} = \left[\begin{array}{cccc} \rho u, & \rho u^2 + p, & \rho uv, & u\left(E + p\right) \end{array} \right]^T, \tag{3a}$$

$$\mathbf{G} = \left[\begin{array}{cccc} \rho v, & \rho uv, & \rho v^2 + p, & v\left(E + p\right) \end{array} \right]^T, \tag{3b}$$

the geometric source terms are

$$\mathbf{S_r} = \frac{2}{r}\left[\begin{array}{cccc} \rho u, & \rho u^2, & \rho uv, & u\left(E+p\right) \end{array}\right]^T, \tag{4a}$$

$$\mathbf{S}_\theta = \frac{1}{r\tan\theta}\left[\begin{array}{cccc} \rho v, & \rho uv, & \rho v^2, & v\left(E+p\right) \end{array}\right]^T, \tag{4b}$$

and the source terms due to gravity are

$$\mathbf{D} = \left[\begin{array}{cccc} 0, & -\rho g\cos\theta, & \rho g\sin\theta, & \rho g\left(u\cos\theta - v\sin\theta\right) \end{array}\right]^T. \tag{5}$$

The symbols ρ, u, v, E and p are the density, radial and polar velocity components, total energy and pressure, respectively, whilst g is the acceleration due to gravity. Equation 1 is closed with a stiffened gas equation of state

$$p = \left(\gamma - 1\right)\left(E - \frac{1}{2}\rho\left|\mathbf{u}\right|^2\right) - \gamma p_c, \tag{6}$$

where $\mathbf{u}$ is the velocity vector, γ is the ratio of specific heats and p_c is a constant. For air, $\gamma = 1.4$ and $p_c = 0$, in which case equation 6 collapses to the ideal gas equation of state. For water, I use $\gamma = 7$ and $p_c = 3 \times 10^8$Pa.

Whilst Ω is spherical, the constraint of polar axisymmetry limits the region of computation to the set of points $\mathbf{r} \in \left([0, R_D], [0, \pi]\right)$. The domain is discretised into $m \times n$ cells, with side length and divergence angle δr and $\delta\theta$ respectively.

All motion external to Ω is assumed to be radial and spherically symmetric. As the water is inviscid and initially stationary, the velocity field external to Ω may be described by a velocity potential ϕ, which is assumed to obey the linear acoustic wave equation, with a general solution $\phi = \frac{1}{r}f\left(t - \frac{r}{c}\right)$. Passing this solution back into equation 1, after some manipulation, the following expressions (which I give here as I use them later in the paper) are obtained

$$u = \frac{f}{r^2} + \frac{f'}{rc}, \tag{7}$$

$$f = r^2 \left(u - \frac{h}{c} - \frac{u^2}{2c} \right), \tag{8}$$

$$f' = r \left(h + \frac{u^2}{2} \right) \tag{9}$$

and

$$f'' = rc\frac{\partial u}{\partial t} - ch - \frac{cu^2}{2}, \tag{10}$$

where c is the speed of sound, and the enthalpy h is approximated by $h = (p - p_\infty)/\rho_\infty$, the subscript ∞ denotes the undisturbed solution in the far field at $t = 0$. Properties with subscript ∞ are set locally according to the initial properties on Γ. The argument of f, $t - r/c$, has been dropped for ease of writing, and a prime denotes differentiation with respect to the argument.

After further manipulation of equations 7 to 10, and substitution into equation 1, the following expressions are obtained

$$\left(\frac{\partial u}{\partial r} \right)_{NLAA} = \frac{u}{r} \left(\frac{u}{2c} - 2 \right) + \frac{(p - p_\infty)}{\rho rc} - \frac{1}{c}\frac{\partial u}{\partial t}, \tag{11}$$

$$\left(\frac{\partial p}{\partial r} \right)_{NLAA} = \frac{\rho u^2}{r} \left(2 - \frac{u}{2c} \right) - \frac{u\,(p - p_\infty)}{rc} + \rho \left(\frac{u}{c} - 1 \right) \frac{\partial u}{\partial t}, \tag{12}$$

$$\left(\frac{\partial \rho}{\partial r} \right)_{NLAA} = 0. \tag{13}$$

If the properties ρ, u and p are those on Γ, then the derivatives obtained from equations 11, 12 and 13 are the spatial derivatives on Γ of a solution which obeys the NLAA.

The boundary condition is applied following the widely used characteristic boundary condition formulation, developed by Thompson (1987, 1990). Within Ω the solution is determined numerically. External to Ω, the solution is approximated by the non-linear acoustic approximation, based on the solution on Γ. The derivatives obtained from equations 11, 12 and 13 are used to define any characteristic waves incoming to Ω on Γ. The solution is updated on the boundary by integrating the characteristic form of the Euler

equations. King et al. (2015) demonstrated the efficacy of this method in one and two spatial dimensions.

INCLUDING THE EFFECTS OF THE SEA SURFACE

To include the sea surface, a virtual image of the air gun is introduced, which is a reflection of the gun in the sea surface. This virtual image is referred to as the "ghost", and signals originating from the ghost are referred to as "ghost waves". Figure 1 shows this configuration. I define a ghost domain $\hat{\Omega}$, bounded by $\hat{\Gamma}$, which is a reflection of Ω in the sea surface, such that it is centred on the point $(2d,\, 0)$, where d is the depth of the gun below the sea surface. Note that whilst the depth of the bubble changes as the bubble rises, Ω and $\hat{\Omega}$ are stationary, and the bubble moves within the domain. The position vector in a coordinate system based at the centre of $\hat{\Omega}$ is $\hat{\mathbf{r}} = \left(\hat{r}, \hat{\theta}\right)$, as shown in Figure 1. Note that the reflection of the domain leads the coordinate system based in $\hat{\Omega}$ to be left-handed, and that should the method be extended to three-dimensional simulations, care must be taken in this respect. Let the solution within Ω be $\mathbf{U}$, and the solution within $\hat{\Omega}$ be $\hat{\mathbf{U}}$. The solution within $\hat{\Omega}$ is assumed to behave identically to the solution within Ω, such that $\hat{\mathbf{U}}\left(\hat{r}, \hat{\theta}, t\right) = \mathbf{U}\left(r, \theta, t\right)$. The sea surface reflection coefficient of -1 is applied later.

The NLAA may be used to calculate the velocity and pressure anywhere outside Ω based on the past solution on Γ. I denote the approximate solution calculated with the NLAA, based on the solution $\mathbf{U}$ on Γ, $\mathbf{U}_\Gamma$, as $\mathbf{U}_{NLAA}$. The approximate solution calculated with the NLAA, based on the solution $\hat{\mathbf{U}}$ on $\hat{\Gamma}$, $\hat{\mathbf{U}}_{\hat{\Gamma}}$, is $\hat{\mathbf{U}}_{NLAA}$. I modify the NLAA boundary condition such that the solution external to Ω and $\hat{\Omega}$ is composed of the superposition of $\mathbf{U}_{NLAA}$ and $\hat{\mathbf{U}}_{NLAA}$. This external solution is used to define characteristics incoming to Ω, and so applying a boundary condition on Γ. In this way the waves due to the ghost are

transmitted into Ω, where they subsequently interact with the bubble.

The method for including the ghost in the NLAA boundary condition is summarised as follows. For each point A on Γ

1. find the shortest distance between $\hat{\Omega}$ and A;
2. use this distance to provide a delay τ, based on the constant and uniform speed of sound in the water, which is the time required for signals due to the ghost to reach A;
3. search the past solution on Γ to find the solution at $t - \tau$;
4. use the NLAA to calculate the pressure and velocity, and their derivatives, at A due to signals emitted from $\hat{\Omega}$ - this involves a non-linear scaling of the past solution;
5. use a geometrical transform to rotate the velocities and derivatives into a frame of reference oriented with the coordinate system in Ω;
6. augment the NLAA boundary condition described in King et al. (2015) with these pressures, velocities and derivatives due to $\hat{\Omega}$.

I now give a detailed description of the method. The boundary condition is applied in this way for every computational cell on Γ. I describe the implementation in a cell A, which lies on Γ. Let the coordinates of cell A be $\mathbf{r_A} = (r, \theta)$. As A is on Γ, $r = R_D$, although for clarity of exposition this substitution is not made in the following derivation. Denote the coordinates of cell A in the coordinate system of $\hat{\Omega}$ as $\hat{\mathbf{r}}_\mathbf{A} = \left(\hat{r}, \hat{\theta}\right)$, as shown in Figure 1. The unit vectors in the ghost-centered coordinate system are $\mathbf{e}_{\hat{\mathbf{r}}}$ and $\mathbf{e}_{\hat{\theta}}$. Simple

pythagorean geometry leads to expressions for $\hat{r}$ and $\hat{\theta}$,

$$\hat{r} = \left(4d^2 - 4dr\cos\theta + r^2\right)^{\frac{1}{2}}, \tag{14a}$$

$$\hat{\theta} = \tan^{-1}\left(\frac{\sin\theta}{\left(\frac{2d}{r} - \cos\theta\right)}\right). \tag{14b}$$

Label as cell B the cell which contains the point at which a line between $(2d, 0)$ and A intersects $\hat{\Gamma}$, as in Figure 1. If $d \gg R_D$, $\hat{\theta}$ is small, and $\frac{\partial\hat{\theta}}{\partial\theta}$ is also small, such that in some cases the index of cell B will not change as θ varies between 0 and π. Note that for $d/R_D = 7.7$ (a value investigated in Numerical Results), the maximum value of $\hat{\theta}$ is approximately 0.065 radians, or 3.7 degrees. In the present scheme, where $\delta\theta = \pi/50$ radians, the index of cell B varies only between two adjacent cells as θ varies between 0 and π. The time taken for signals to travel from B to A is $\tau = (\hat{r} - r)/c$, where c is the speed of sound used in the NLAA calculations, based on the density and pressure at A. Let $t_G = t - \tau$, and denote the properties $(\cdot)$ at B at time t_G, $(\cdot)_B$. Then ρ_B, u_B, p_B and $\frac{\partial u_B}{\partial t}$ are used to calculate the wavefunction f and its derivatives f' and f'', using equations 8, 9 and 10

$$f = r^2\left(u_B - \frac{p_B - p_\infty}{\rho_\infty c} - \frac{u_B^2}{2c}\right), \tag{15a}$$

$$f' = r\left(\frac{p_B - p_\infty}{\rho_\infty} + \frac{u_B^2}{2}\right), \tag{15b}$$

$$f'' = rc\frac{\partial u_B}{\partial t} - c\frac{p_B - p_\infty}{\rho_\infty} - \frac{cu_B^2}{2}. \tag{15c}$$

Note that this step is actually done separately. At each time step, f, f' and f'' are calculated for all B required. This is the only historical part of the solution which is stored. This approach minimises the storage requirements of the code. If $t_G < 0$, the signal due to the ghost cannot have reached Ω. In this case, the NLAA boundary condition is simply applied in the usual manner, as described in King et al. (2015), whilst f, f' and f'' are stored for

use once $t_G \geq 0$. Note that when searching the past solution for the solution at t_G, I use the nearest sample. A better method would be to interpolate. However, as the time step in the present scheme is typically of the order of 10^{-6} the error introduced by this approach is small, and in any case the consequence will only be the introduction of a small amount of dissipation in the ghost wave which is transmitted into the domain.

The NLAA is used to estimate the solution at A due to the past solution at B. Let $\hat{u}_G$, p_G, $\frac{\partial \hat{u}_G}{\partial \hat{r}}$ and $\frac{\partial p_G}{\partial \hat{r}}$ be the velocity component in the direction $\mathbf{e}_{\hat{\mathbf{r}}}$ due to the ghost, the pressure due to the ghost, and their derivatives, at A. Taking equations 7, 9 and the derivatives thereof, and substituting into equation 1, they are given by

$$\hat{u}_G = R_{ss} \left\{ \frac{f}{\hat{r}^2} + \frac{f'}{\hat{r}c^2} \right\}, \tag{16}$$

$$p_G = R_{ss}\rho_\infty \left(\frac{f'}{\hat{r}} - \frac{\hat{u}_G^2}{2} \right) + p_\infty, \tag{17}$$

$$\frac{\partial \hat{u}_G}{\partial \hat{r}} = R_{ss} \left\{ \frac{-2f}{\hat{r}^3} - \frac{2f'}{\hat{r}^2 c} - \frac{f''}{\hat{r}c^2} \right\} \tag{18}$$

and

$$\frac{\partial p_G}{\partial \hat{r}} = -R_{ss}\rho_\infty \left\{ \hat{u}_G \frac{\partial \hat{u}}{\partial \hat{r}} + \frac{f'}{\hat{r}^2} + \frac{f''}{\hat{r}c} \right\}, \tag{19}$$

where R_{ss} is the reflection coefficient of the sea surface, and $R_{ss} = -1$. If the sea surface were replaced with a rigid boundary, this would be accounted for in the model by setting $R_{ss} = 1$. I neglect $\hat{v}_G$, $\frac{\partial \hat{u}_G}{\partial \hat{\theta}}$ and $\frac{\partial p_G}{\partial \hat{\theta}}$, as the NLAA is based on the assumption of spherical symmetry. Let $\psi = \pi - \hat{\theta} - \theta$. The components of the velocity field due to the ghost in the coordinate system centred on the real gun can be calculated using the rotation $\mathbf{u}_G = \mathbf{M}\,(\hat{u}_G,\, 0)^T$, where $\mathbf{u}_G = u_G \mathbf{e}_\mathbf{r} + v_G \mathbf{e}_\theta$ and

$$\mathbf{M} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}, \tag{20}$$

giving

$$u_G = \hat{u}_G \cos\psi \quad \text{and} \quad v_G = \hat{u}_G \sin\psi. \tag{21}$$

The derivatives of $\hat{u}_G$ and p_G in the coordinate system centred on the real gun can be obtained, again using the rotation $\mathbf{M}$, as $\nabla(.) = \mathbf{M}\hat{\nabla}(.)$, where $\nabla(.) = \frac{\partial(.)}{\partial r}\mathbf{e_r} + \frac{1}{r}\frac{\partial(.)}{\partial\theta}\mathbf{e}_\theta$ and $\hat{\nabla}(.) = \frac{\partial(.)}{\partial\hat{r}}\mathbf{e_{\hat{r}}} + \frac{1}{\hat{r}}\frac{\partial(.)}{\partial\hat{\theta}}\mathbf{e}_{\hat{\theta}}$. Hence the derivatives of p_G are given by

$$\frac{\partial p_G}{\partial r} = \frac{\partial p_G}{\partial \hat{r}}\cos\psi, \tag{22a}$$

$$\frac{\partial p_G}{\partial \theta} = r\frac{\partial p_G}{\partial \hat{r}}\sin\psi, \tag{22b}$$

and the derivatives of $\hat{u}_G$ are

$$\frac{\partial \hat{u}_G}{\partial r} = \frac{\partial \hat{u}_G}{\partial \hat{r}}\cos\psi, \tag{23a}$$

$$\frac{\partial \hat{u}_G}{\partial \theta} = r\frac{\partial \hat{u}_G}{\partial \hat{r}}\sin\psi. \tag{23b}$$

If $2d \gg R_D$, the variation of $\hat{\theta}$ with r and θ can be neglected, allowing the approximation that $\frac{\partial\cos\psi}{\partial r} = \frac{\partial\sin\psi}{\partial r} = 0$, $\frac{\partial\sin\psi}{\partial\theta} = -\cos\psi$ and $\frac{\partial\cos\psi}{\partial\theta} = \sin\psi$. The errors introduced by this approximation vary around Γ. The maximum relative value of these errors is of the order of $R_D/2d$. The derivatives of the velocity field due to the ghost are obtained using equations 21 and 23

$$\frac{\partial u_G}{\partial r} = \frac{\partial \hat{u}}{\partial \hat{r}}\cos^2\psi, \tag{24a}$$

$$\frac{\partial u_G}{\partial \theta} = r\frac{\partial \hat{u}}{\partial \hat{r}}\sin\psi\cos\psi + \hat{u}\sin\psi, \tag{24b}$$

$$\frac{\partial v_G}{\partial r} = \frac{\partial \hat{u}}{\partial \hat{r}}\cos\psi\sin\psi, \tag{24c}$$

$$\frac{\partial v_G}{\partial \theta} = r\frac{\partial \hat{u}}{\partial \hat{r}}\sin^2\psi - \hat{u}\cos\psi. \tag{24d}$$

The boundary condition is now applied by augmenting equations 11 and 12 with the derivatives obtained from equations 22 and 24 giving

$$\left(\frac{\partial u}{\partial r}\right)_{NLAA} = \frac{u}{r}\left(\frac{u}{2c} - 2\right) + \frac{(p - p_\infty)}{\rho r c} - \frac{1}{c}\frac{\partial u}{\partial t} + \frac{\partial u_G}{\partial r}, \tag{25}$$

$$\left(\frac{\partial p}{\partial r}\right)_{NLAA} = \frac{\rho u^2}{r}\left(2 - \frac{u}{2c}\right) - \frac{u\left(p - p_\infty\right)}{rc} + \rho\left(\frac{u}{c} - 1\right)\frac{\partial u}{\partial t} + \frac{\partial p_G}{\partial r}. \tag{26}$$

I apply the boundary condition in the same manner as in King et al. (2015), using the characteristic boundary condition formulation of Thompson (1990), and equations 25, 26 and 13 to define any incoming characteristics on Γ. All that remains is to modulate p_∞ with the pressure field due to the ghost, according to

$$p_\infty = p_\infty\left(t = 0\right) + \frac{R}{\rho_\infty}\left(\frac{f'}{\hat{r}} - \frac{\hat{u}_G^2}{2}\right). \tag{27}$$

This completes the process by which the ghost is included. All remaining aspects of the model are unaltered.

The above method is described for a virtual gun located at $(2d, 0)$. However, with minor changes to the geometry, the method could be extended for a virtual gun, or indeed another real gun, at any location, or multiple real and virtual guns. Note that the imposition of axisymmetry limits the present model. A ghost gun located at $(2d, \pi/2)$ would actually approximate a toroidal bubble with major radius d. The solution in $\hat{\Omega}$ is not required to mirror the solution in Ω. The two domains could be separate finite volume simulations of bubbles, and the above method would allow them to interact, without the need to simulate the entire region between the two bubbles. With three-dimensional finite volume models, this approach could be used to simulate entire arrays of air guns, including the effects of the sea surface. However, this approach would significantly increase the computational costs.

NUMERICAL SCHEME

The numerical scheme used in this paper is the same as that described in King et al. (2015), but with the modifications to the boundary condition described above. In this section I

give a brief outline of the numerical scheme. Full details of the scheme are in King et al. (2015).

The numerical scheme is constructed from a single-phase Euler solver in conjunction with a ghost fluid method (GFM) (Fedkiw et al., 1999; Wang et al., 2006) to model the interface, which is tracked with a level set (Russo and Smereka, 2000). The single-phase Euler solver is a dimensionally split Godunov-type scheme (Godunov, 1959; Einfeldt, 1988; Ivings et al., 1998), which is first-order accurate in space and time. Numerical fluxes (**F** and **G** in equation 1) are constructed using the HLLC flux (Toro et al., 1994) and are calculated using an approximate Roe-type Riemann solver due to Hu et al. (2009). A first-order operator splitting approach is used for dimensional splitting, and also to included the geometric and gravitational source terms $\mathbf{S_r}$, $\mathbf{S}_\theta$ and $\mathbf{D}$, in equation 1. I note that the word "ghost" in ghost fluid method has nothing to do with the ghost being considered in this paper. The ghost fluid method is used to simulate the interface between two immiscible fluids, and the term "ghost" refers to the way in which the GFM splits the computational domain into "real" and "ghost" cells.

Model parameters and initial conditions

The initial conditions are given by

$$(\rho,\, \mathbf{u},\, p,\, \gamma,\, p_c) = \begin{cases} \big(102,\, 0,\, 8.85\times10^6 + p_h,\, 1.4,\, 0\big) & \text{if } 0 \le r \le 0.1, \\ \big(\rho_w,\, 0,\, p_w,\, 7.0,\, 3\times10^8\big) & \text{if } 0.1 < r \le R_D, \end{cases} \tag{28}$$

where $\rho_w = 1000$, $p_w = p_{atm} + \rho_w g d + p_h$, $p_h = -\rho g r \cos\theta$, $p_{atm} = 1.01325 \times 10^5$ and d is the depth of the gun. All quantities are in S.I. Units, such that radius, density, velocity

and pressure have units of m, kgm^{-3}, ms^{-1} and Pa, respectively. P_c has units of Pa, and γ is dimensionless. In all cases, $g = 9.81\text{ms}^{-2}$, the CFL number (Courant et al., 1928, 1967) is 0.8 and $\delta\theta = \pi/50$ radians. Except where explicitly stated otherwise, $\delta r = 0.02$m and $R_D = 1$m. p_∞ is initialised locally on Γ to p_w.

This set of initial conditions aims to simulate a 250 cubic inch (4.01 litres) air gun, fired at a pressure of 2000 psi (13.79 MPa). Note that the discrepancy in initial pressures - $8.85 \times 10^6\text{Pa} \approx 1300\text{psi}$ - is intentional, and is designed to account for the process by which air is released from the gun. I note here that my primary aim is to capture qualitatively the physics of the ghost-bubble interaction, in a more realistic way than has been acheived in previous models. My primary aim is not to acheive a good match with measured data. If this were my aim, a more traditional type of model such as, for example, that of Ziolkowski and Metselaar (1984) would perform better.

NUMERICAL RESULTS

I run the simulation with and without the ghost, for gun depths of $d = 11.5$, 9.5, 7.7, 6.2, 5.5, 5, 4.5, 4, 3.5, 3 and 2.5 metres. I calculate an effective bubble radius, referred to as the interface position or R_{int}, as the radius of a sphere with the same volume as the bubble. I calculate the interface pressure P_{int} as the average pressure over the bubble surface.

The effect of the ghost on bubble oscillations

Figure 2 shows the time variation of R_{int} and P_{int} without the ghost (dashed lines), and with the ghost (solid lines) for bubbles with depths $d = 11.5, 7.7, 5$ and 3 metres. The influence of the ghost on R_{int} and P_{int} is small for $d = 7.7$ and 11.5 metres. As d is reduced,

the influence of the ghost becomes more significant. In all cases, the effect of the ghost is to increase the maximum bubble radius $R_{\max}$ and to reduce the minimum value of R_{int} reached during collapse. The maximum pressure in the bubble during collapse $P_{collapse}$ is also increased due to the ghost, and the bubble period $t_{collapse}$ is reduced. As the wave due to the ghost passes the bubble, the pressure surrounding the bubble is reduced. This reduces the effective stiffness of the water to the expanding bubble, causing the larger $R_{\max}$ and the stronger, sharper collapse observed in Figure 2. In all cases the simulation breaks down after the first bubble period. This is a consequence of the sensitivity of the bubble shape during collapse, and of the instability inherent in axisymmetric polar coordinate systems, which leads to axial jetting along the polar axis (Kifonidis et al. (2006); King et al. (2015)) causing the simulation to break down.

Figure 3 shows the relative changes in bubble properties due to the ghost, $\Delta(\cdot)_{rel}$ for $(\cdot) = R_{\max}$ and $(\cdot) = -t_{collapse}$. Note the change of sign for the changes in bubble period. The relative changes are calculated according to $\Delta(\cdot)_{rel} = [(\cdot)_G - (\cdot)_{NG}] / (\cdot)_{NG}$, and a subscript G indicates the ghost is included, whilst a subscript NG indicates the ghost is not included.

For bubbles deeper than approximately 5 metres, the effect of the ghost on the maximum bubble radius scales inversely with the depth; $\Delta R_{\max,rel}$ is proportional to $1/d$. Shallower bubbles do not follow this trend. This may be due to the increasing damage done by the approximation that $2d \gg R_D$ as d is reduced. The changes in the bubble period due to the ghost show a similar trend. However, the curve in Figure 3 for $\Delta t_{collapse,rel}$ shows much more variation around this trend. This is a consequence of the physical instability of the bubble during the late stages of collapse, which leads to a sensitivity of bubble properties on collapse. Similarly, the relative change in peak interface pressure on collapse $P_{collapse}$

due to the ghost shows significant variation. For all depths simulated the ghost causes an increase in $P_{collapse}$, although this increase varies between 3% and 40% and does not follow any clear trend.

The effect of the ghost on signals in the near field

Figure 4a shows the variation of pressure at the point $(r, \theta) = (R_D, \pi/2)$, henceforth referred to as $p_{R_D,\pi/2}$, with time, without the ghost (dashed lines), and with the ghost (solid lines) for $d = 11.5, 7.7, 5$ and 3 metres. Figure 4b shows the difference in $p_{R_D,\pi/2}$ with and without the ghost (ie. $p_{R_D,\pi/2,G} - p_{R_D,\pi/2,NG}$), for $d = 11.5, 7.7, 5$ and 3 metres. The time at which the ghost begins to affect $p_{R_D,\pi/2}$ is proportional to d. The magnitude of the first disturbance to $p_{R_D,\pi/2}$ due to the ghost is proportional to $1/d$, which is to be expected, as, according to the NLAA, the pressure around the bubble scales inversely with the distance from the bubble in regions where the distance from the bubble is great enough that the velocity terms are small.

The steps in all traces in Figure 4a are due to the ghost fluid method (GFM), which is used to simulate the air-water interface. As the interface passes the centre of each cell, the set of cells which are interface cells changes, and so the conditions which control the transmission of energy and momentum across the interface change. This causes a small pressure pulse to propagate outwards from the interface. The notches after $t = 0.02$ seconds in all traces in Figure 4b are due to this aspect of the GFM. As the ghost causes a slightly different bubble expansion rate, the times at which the interface passes a cell centre change, which lead to the notches visible in Figure 4b. These notches are not caused by the application of the boundary condition. These steps and notches in the results are numerical

artefacts of the discretisation. These errors are reduced by mesh refinement.

Figure 4a clearly shows a notch in the radiated pressure as the ghost wave passes. The ghost wave is calculated based on the past solution on Γ. The outgoing wave which impacts on Γ has been propagated across the domain, and due to the numerical viscosity inherent in the scheme, is slightly smeared. Hence the ghost wave also appears slightly smoother than in other models where there is no spatial discretisation, for example those of Ziolkowski (1970) and Cox et al. (2004). The incoming ghost wave has the same form as the outgoing initial pulse, but scaled and inverted. However, the near-field pressures shown in Figures 4a and 4b do not show the ghost wave with this form. In this respect my results differ from those of Ziolkowski (1970) and Cox et al. (2004). This is because the pressure which is recorded at $(R_D, \pi/2)$ is a combination of the wave which has travelled directly from the ghost to the point $(R_D, \pi/2)$, and the wave due to the ghost which has reflected off the bubble, a phenomenon which has not been captured in previous models.

When the ghost wave impacts on the bubble it is both transmitted into the bubble and reflected off the bubble surface, with a reflection coefficient close to -1. The extra distance travelled by the wave which reflects off the bubble when it reaches $(R_D, \pi/2)$ is of the order of R_D. The time between the trough, or first arrival (marked 'a' for $d = 3$ metres on Figure 4b), and the peak, or second arrival (marked 'b' for $d = 3$ metres on Figure 4b) of each trace in Figure 4b is close to R_D/c_w, where c_w is the speed of sound in water, which is evidence of this explanation. For $d = 3$ metres, this delay is 0.71ms, whilst $R_D/c_w \approx 0.67$ms. The signals produced by this mechanism ought to have a period of approximately $2R_{int}/c_b$, where c_b is the speed of sound in the bubble. For a bubble with $R_{int} = 0.27$ metres when the ghost wave arrives, and a speed of sound of approximately $c_b = 230\text{m}^{-1}$ (values corresponding to the results for $d = 3$ metres), this gives an expected

period of 2.3ms, and hence frequencies around 425Hz. The results shown in this paper use a fairly dissipative scheme (in order to reduce computational costs). However, in Figure 4b the trace for $d = 3$ shows approximately this period (actually 2.48ms) between the second arrival (marked 'b') and the second reflection (marked 'c' on Figure 4b). There is further evidence of this phenomenon visible in Figure 7, which I discuss later.

The shape of the bubble

As mentioned above, towards the end of collapse, the shape of the bubble is sensitive to small disturbances. Figure 5 shows the shape of the bubble during collapse. The bubble is initialised with $d = 7.7$ metres. The dashed line shows the shape of the bubble without the ghost, and the solid line shows the shape with the ghost. In frame (a), during the early stages of collapse, the bubble shapes are similar, although the bubble with the ghost is slightly larger than that without the ghost. There are slight differences in shape visible in frame (a), and with the ghost included the bubble is slightly less symmetrical. As the bubbles collapse, their shapes begin to differ more, as the slight variations in shape are amplified, and by the end of collapse (frame (f)) there are significant differences between the two bubble shapes. The main aspect of the ghost signal which influences bubble motion is the initial pulse of low pressure, which for $2d \gg R_D$ is close to planar, and passes through the bubble from above. It might be expected that this would cause the bubble to rise slightly, and for the upper surface to expand more rapidly than the lower surface. At the instant of impact, the speed of sound in the water is nearly an order of magnitude greater than the speed of sound in the air. Hence, the ghost wave travels round the bubble far faster than it propagates into the bubble, and so the asymmetric effect of the ghost wave on the bubble is reduced.

The impact of the ghost

Figure 6 shows the initial impact of the signal due to the ghost on the bubble, with $d = 7.7$ metres. To obtain these results, the model is run both with and without the ghost. The results without the ghost are subtracted from the results with the ghost to obtain the difference. In this way, the ghost wave can be seen. Let the subscript 'G' refer to results with the ghost present, and the subscript 'NG' to those without the ghost. Figure 6 shows the quantity $||\mathbf{u}_G| - |\mathbf{u}_{NG}||$ at various times as the ghost wave impacts on the bubble.

Frame (a) shows the bubble just prior to the impact of the ghost wave. In frame (b) the ghost wave has just reached the bubble, and has begun to propagate both through and around it. In frame (c) the ghost wave has propagated round the bubble, and is propagating into the bubble, as can be seen by the bright crescent in the upper half of the bubble. The difference in sound speeds within the bubble and in the water can now be clearly seen. Whilst the wave has travelled round the bubble and beyond in the water, a distance of more than two bubble radii, within the bubble it has not yet passed the origin, having travelled about 0.4 bubble radii. In frame (d) the ghost wave continues to propagate through the bubble, whilst in the water it has passed the bubble and continues out through the domain boundary. The high sound speed in the water compared with the air means that the wave propagating into the bubble is nearly spherical, and not planar. The bubble focusses the ghost wave on the origin, from where it is reflected, as described in the previous sections.

As soon as the ghost wave impinges on the domain it affects the value of the time step δt. In order to compare the results with and without the ghost, and to avoid complex interpolations, for these results I force the time step to remain constant for $t > 10$ms.

Despite this strategy, there are numerical artefacts present in Figure 6. These are visible as the two faint diagonal lines through the ghost wave in frames (a) to (c) of Figure 6. These are due to the subtraction of one field from another. The change in pressure due to the ghost causes small differences in the bulk motion in and around the bubble, and when one set of results is subtracted from the other, this leads to the numerical artefacts mentioned above. This analysis can only be conducted at this stage. If conducted after the ghost has significantly changed the shape of the bubble and the flow field surrounding and within it, this analyis would give relatively meaningless results.

Finally I run the simulation at a higher resolution in the radial direction, with $\delta r = 0.01$, for $d = 7.7$ metres, both with and without the ghost. Figure 7 shows the impact of the ghost wave on the bubble in this case. Outside the bubble, the intensity of the image corresponds to the magnitude of the difference in pressure with and without the ghost, $|p_G - p_{NG}|$. Due to the difference in equations of state for air and water, the pressure differences within the bubble are small compared with those outside, and are not visible if the same scale is used. Within the bubble, contours of the difference in velocity magnitude are shown. Whilst there are no values on these contours, they clearly show the shape of the ghost wave propagating through the bubble.

Figure 7 shows several interesting features. The numerical artefacts of subtraction are clearly visible around the bubble - in the form of speckled regions - in all frames of Figure 7. These are not due to errors in the code, but are simply an artefact of the method I employ to visualise the results. The diagonal line in frame (a) spreads round the domain as the ghost wave passes through. This line appears to originate from the boundary, and I believe it to be an error introduced by the boundary condition. However, as the bubble is expanding when the ghost wave impacts, this error seems to be advected back out of the domain before

it affects the bubble. Most importantly, Figure 7 shows the reflection of the ghost wave off the bubble. In frame (a) the ghost wave is just impacting on the bubble. In frame (b) the ghost wave has propagated a small distance into the bubble, and a reflection of the ghost wave (with a reflection coefficient of approximately -1) from the upper surface of the bubble is visible as the red patch above the bubble. This is even clearer a short time later in frame (c). Frame (d) shows this effect when the ghost wave has travalled most of the way round the bubble. It is this reflection which causes the second arrival in the near-field pressure marked 'b' for $d = 3$ metres on Figure 4b.

As a sound wave impacts on the bubble, it is partially transmitted through the interface, and partially reflected. The high speed of sound in the water compared with the bubble causes the bubble to focus the ghost wave on the centre of the bubble. It is then reflected, and when it reaches the bubble surface it is again partially reflected and partially transmitted into the water. The process repeats, with diminishing amplitude, leading to higher frequencies in the pressure field around the bubble as the ghost wave passes the bubble (the first of these higher frequency pulses is marked 'c' for $d = 3$ metres in Figure 4b. Figure 8 shows the solution a short time after the ghost wave has impacted on the bubble. The quantities plotted in Figure 8 are the same as those plotted in Figure 7. Concentric pressure waves which are nearly spherical can be seen propagating back and forth through the bubble. Every time these impact on the bubble surface, they are partially transmitted out into the water, as can be faintly seen in Figure 8.

All these effects of the asymmetric bubble-ghost interaction have not been simulated in previous air gun modelling, and provide a potential explanation for the high frequencies observed as the ghost wave impacts the bubble in experimental measurements. However, it must be borne in mind that, in reality, the interaction is further complicated by the presence

of the gun within the bubble.

EXPERIMENTAL OBSERVATIONS

In the previous section I describe a mechanism for bubble-ghost interactions not included in previous models. In this section I show measurements that support my theory. As a part of a Joint Industry Program (JIP), Petroleum Geo-Services (PGS) conducted the Svein Vaage broadband air gun study (Mattssonn et al., 2012) between June and October 2007 and again between June 2009 and June 2010. In this study single air guns and air gun clusters were fired at a test barge in a fjord on the west coast of Norway, whilst near-field and far-field signals were recorded. For the portion of data I analyse, a single 250 cubic inch air gun was suspended at a depth of 6 metres and fired at 2000psi. A near-field hydrophone located a distance of 1.5 metres from the gun ports and at the same depth was used to record the signal. The data were recorded with a sample rate of 100kHz. I analyse a single shot, although the trends I observe are present in all shots with this configuration.

Figure 9a shows the first 150ms of the signal recorded at the hydrophone. Note the ghost reflection at approximately $t = 0.014$ seconds, followed by small ripples until about $t = 0.04$ seconds. Figure 9b shows the amplitude spectrum of this first 150ms of the signal. The small peak marked 'a' in Figure 9b corresponds to frequencies between 400 and 600Hz, and is due to the ripples after the impact of the ghost wave.

Figure 10a shows a portion of the data from Figure 9a, around the time at which the ghost wave first reaches the hydrophone. The point marked 'a' corresponds to the first arrival of the ghost at the hydrophone. The point marked 'b' corresponds to the arrival of the first reflection of the ghost wave off the bubble surface. The ripples, the first two

of which are marked ‘c’ and ’d’ correspond to the oscillation of the ghost wave within the bubble. These ripples continue in a slightly uneven manner, and with decreasing amplitude until about $t = 0.04$ seconds. The data in Figure 10a are in good qualitative agreement with the theory developed in the previous sections, based on the numerical results shown in Figures 4, 7 and 8.

I calculate the average slope of the data in Figure 10a between approximately the points marked ‘b’ and ‘e’. By subtracting a straight line with this slope from the measured values, and then subtracting the average of the result, I obtain an approximate image of these ripples due to the ghost. I show these in Figure 10b. The points marked on Figure 10b correspond to the points marked on Figure 10a. The frequencies of the large amplitude ripples in Figure 10b vary between roughly 350 and 600Hz. The frequencies I observe here match those expected from the theory of ghost-bubble interactions put forward in the previous section.

The ripples in Figure 10b are not clean. There are clearly other frequencies, both higher and lower, present in the data. In reality, the bubble contains a gun, and the presence of this influences the way the bubble focusses the ghost wave, and may introduce other frequencies as the effective size of the bubble differs in different directions. The high frequencies clearly visible in Figure 10b are not present in the data prior to the impact of the ghost. Whilst not the focus of this paper, I note that these high frequencies may be due to rattling of the gun, or the hydrophone, or mechanical resonances in the system as a whole.

CONCLUSIONS

The effects of the sea surface on air gun bubble oscillations are usually included by introducing a virtual image of the bubble reflected in the sea surface, referred to as the ghost. I include the ghost in a finite volume simulation of an air gun bubble through the use of a novel artificial boundary condition, and hence transmit signals due to the ghost into the computational domain, where they interact with the bubble. In theory it would be possible to use this approach to simulate an entire array of guns. An extension of this kind would require three-dimensional modelling, and a separate finite volume model for each bubble. The interactions between the different bubbles would follow the same method as the interaction between each bubble and its ghost. The computational cost of this approach would be significicant.

Results show that the ghost causes slightly reduced damping in bubble oscillations, and an increase in bubble period. For guns deeper than 5 metres, as the depth of the bubble is increased, the effect of the ghost on the maximum bubble radius varies with the inverse of the depth. The ghost strongly influences the final shape of the bubble, due to the instability of the surface during collapse.

When the ghost wave impacts on the bubble it is partially transmitted into the bubble, and partially reflected. The bubble focusses the ghost wave on the centre of the bubble, from where it is reflected. These reflections - and in reality several subsequent reflections - cause high (around 400Hz) frequencies in the near-field pressure wave for roughly 20ms after the impact of the ghost wave on the bubble. Previous air gun models are unable to capture this phenomenon. This theory is supported by near-field measurements, which show oscillations at frequencies associated with these internal bubble oscillations just after

the impact of the ghost.

In theory, near-field measurements showing these oscillations could be used to determine a parameter which encapsulates the size and composition of the bubble. Such measurements might be used to constrain the more simple models used by industry. However, in reality every air gun bubble contains an air gun, cabling and gun housing. These further complicate the problem, and may introduce other vibrations as the ghost impacts the domain. If the effect of the bubble focussing the ghost wave, which then oscillates within the bubble, were included in a homogeneous spherical bubble model of the type currently used by industry, the match between modelled and measured near-field signatures would be improved.

ACKNOWLEDGMENTS

J. R. C. King is funded by PGS. I thank Anton Ziolkowski for many useful discussions, and for helpful comments on the paper. I thank PGS for providing the data.

REFERENCES

Blake, J. R., G. S. Keen, R. P. Tong, and M. Wilson, 1999, Acoustic cavitation: the fluid dynamics of non-spherical bubbles: Philosophical Transactions of the Royal Society of London A, **357**, 251–267.

Courant, R., K. Friedrichs, and H. Lewy, 1928, ber die partiellen differenzengleichungen der mathematischen physik: Mathematische Annalen, **100**, 32–74.

———, 1967, On the partial difference equations of mathematical physics: IBM Journal of Research and Development, **11**, 215–234.

Cox, E., A. Pearson, J. R. Blake, and S. R. Otto, 2004, Comparison of methods for modelling the behaviour of bubbles produced by marine seismic airguns: Geophysical Prospecting, **52**, 451–477.

Dragoset, W. H., 1984, A comprehensive method for evaluating the design of air guns and air gun arrays: The Leading Edge, **3**, 52–61.

Einfeldt, B., 1988, On Godunov-type methods for gas dynamics: SIAM Journal of Numerical Analysis, **25**, 294–318.

Fedkiw, R. P., T. Aslam, B. Merriman, and S. Osher, 1999, A non-oscillatory Eulerian approach to interfaces in multimaterial flows - the ghost fluid method: Journal of Computational Physics, **152**, 457–492.

Gilmore, F. R., 1952, Collapse of a spherical bubble: Technical Report No. 26-4, Hydrodyn. Lab., Calif. Inst. Tech.

Godunov, S. K., 1959, A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics: Matematicheskii Sbornik, **47(89)**, 271–306.

Hu, X. Y., N. A. Adams, and G. Iaccarino, 2009, On the HLLC Riemann solver for interface interaction in compressible multi-fluid flow: Journal of Computational Physics, **228**,

6572–6589.

Ivings, M. J., D. M. Causon, and E. F. Toro, 1998, On Riemann solvers for compressible liquids: International Journal for Numerical Methods in Fluids, **28**, 395–418.

Kifonidis, K., T. Plewa, L. Scheck, H.-T. Janka, and E. Muller, 2006, Non-spherical core collapse supernovae II. The late-time evolution of globally anistropic neutrino-driven explosions and their implications for SN 1987 A: Astronomy and Astrophysics, **453**, 661–678.

King, J. R. C., A. M. Ziolkowski, and M. Ruffert, 2015, Artificial boundary conditions for simulations of oscillating bubbles using the non-linear acoustic approximation: Journal of Computational Physics, **284**, 273–290.

Kirkwood, J. G., and H. A. Bethe, 1942, Progress report on 'The pressure wave produced by an underwater explosion I': Technical Report No. 588, Office of Scientific Research and Development, US Navy.

Lamb, H., 1923, The early stages of submarine explosion: Philosophical Magazine, **45**, 257–265.

Landrø, M., 1992, Modelling of GI gun signatures: Geophysical Prospecting, **40**, 721–747.

Landrø, M., and R. Sollie, 1992, Source signature determination by inversion: Geophysics, **57**, 1633–1640.

Laws, R. M., L. Hatton, and M. Haartsen, 1990, Computer modelling of clustered airguns: First Break, **8**, 331–338.

Lezzi, A., and A. Prosperetti, 1987, Bubble dynamics in a compressible liquid. Part 2. Second-order theory: Journal of Fluid Mechanics, **185**, 289–321.

Mattssonn, A., G. Parkes, and D. Hedgeland, 2012, Svein vaage broadband airgun study, *in* The Effects of Noise on Aquatic Life: Springer New York, volume **730** *of* Advances in

Experimental Medicine and Biology, 469–471.

Parkes, G. E., A. M. Ziolkowski, L. Hatton, and T. Haugland, 1984, The signature of an air gun array: Computation from near-field measurements including interactions - Practical considerations: Geophysics, **48**, 105–111.

Plesset, M. S., 1949, The dynamics of caviation bubbles: Journal of Applied Mechanics, **16**, 277–282.

Prosperetti, A., and A. Lezzi, 1986, Bubble dynamics in a compressible liquid. Part 1. First-order theory: Journal of Fluid Mechanics, **168**, 457–478.

Rayleigh, J. W. S., 1917, On the pressure developed in a liquid during the collapse of a spherical cavity: Philosophical Magazine, **34**, 94–99.

Russo, G., and P. Smereka, 2000, A remark on computing distance functions: Journal of Computational Physics, **163**, 51–67.

Thompson, K. W., 1987, Time dependent boundary conditions for hyperbolic systems: Journal of Computational Physics, **68**, 1–24.

———, 1990, Time dependent boundary conditions for hyperbolic systems, II: Journal of Computational Physics, **89**, 439–461.

Toro, E. F., M. Spruce, and M. Speares, 1994, Restoration of the contact surface in the HLL-Riemann solver: Shock Waves, **4**, 25–34.

Wang, W., T. G. Liu, and B. C. Khoo, 2006, A real ghost fluid method for the simulation of multimedium compressible flow: SIAM Journal of Scientific Computing, **28**, 278–302.

Ziolkowski, A., 1970, A method for calculating the output pressure waveform from an air gun: Geophysical Journal of the Royal Astronomical Society, **21**, 137–161.

———, 1998, Measurement of air-gun bubble oscillations: Geophysics, **63**, 2009–2024.

Ziolkowski, A. M., and G. Metselaar, 1984, The pressure wavefield of an air gun array:

Expanded Abstracts 54th SEG Meeting, Atlanta, 274–276.

Ziolkowski, A. M., G. E. Parkes, L. Hatton, and T. Haugland, 1982, The signature of an air gun array; computation from near-field measurements including interactions: Geophysics, **47**, 1413–1421.

LIST OF FIGURES

ghost. (a) $t = 10.48$ms; (b) $t = 10.63$ms; (c) $t = 10.78$ms; (d)$t = 11.23$ms.

8 Plots showing the bubble after the ghost wave has impacted on the bubble, modelled with $\delta r = 0.01$ metres. The blue line represents the air-water interface. Outside the bubble the intensity of the image corresponds to the magnitude of the difference in pressure with and without the ghost. The contours within the bubble show the difference in velocity magnitude with and without the ghost. (a) $t = 14.53$ms; (b) $t = 15.43$ms; (c) $t = 16.78$ms.

9 Near-field measurements from a 250cubic inch air gun at 6 metres depth. (a) pressure measurement; (b) amplitude spectrum.

10 The portion of the near-field measurement around the time the ghost wave impacts the bubble. (a) pressure measurement; (b) the difference between the direct measurement and a linear variation of pressure with time.

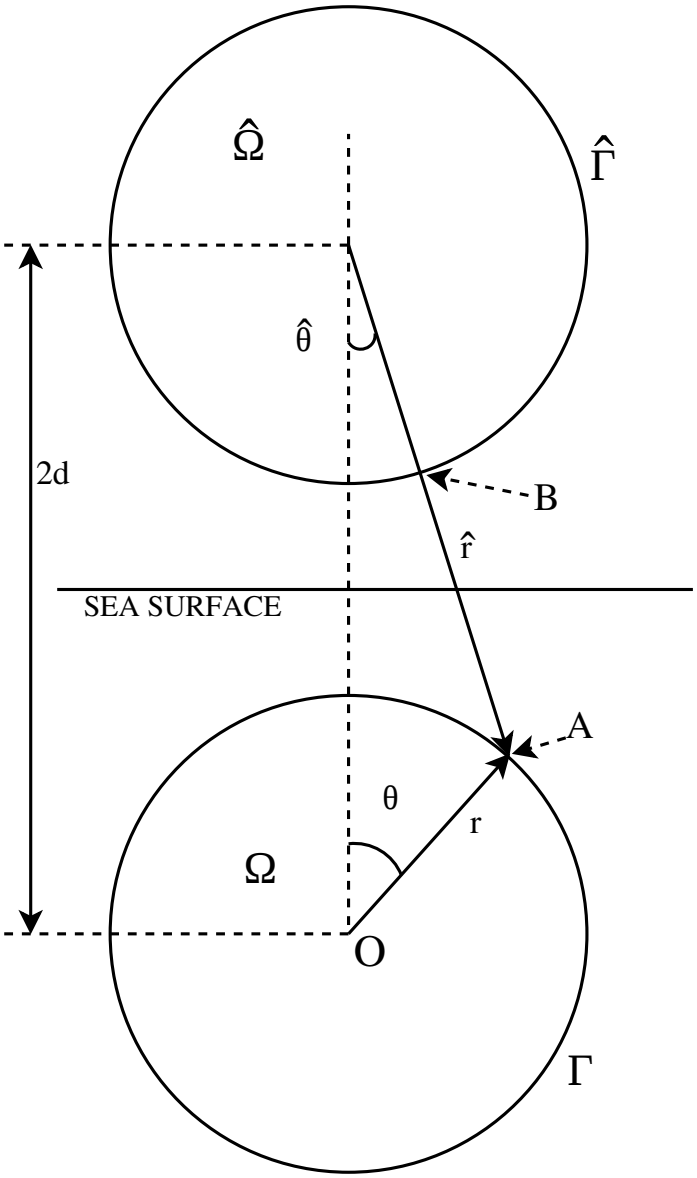
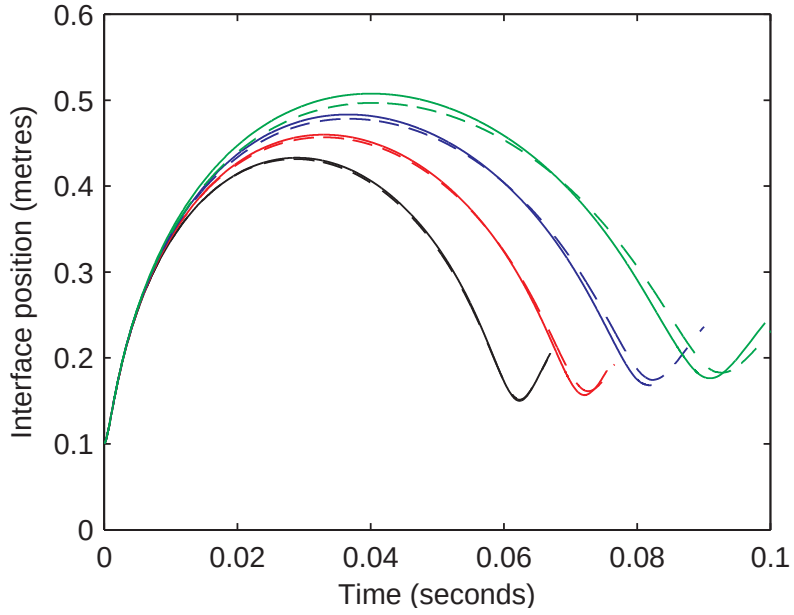
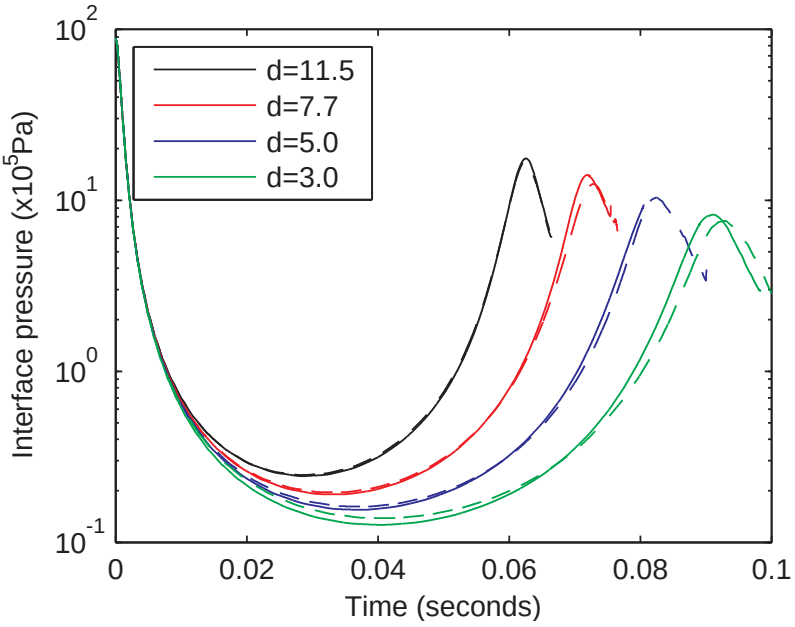


Figure 1: A schematic diagram of the system.

KING –



(a)



(b)

Figure 2: Variation of interface position and pressure with time, without the ghost (dashed lines), and with the ghost (solid lines), for bubbles at depths of $d = 11.5$, 7.7, 5.0 and 3.0 metres.

KING –

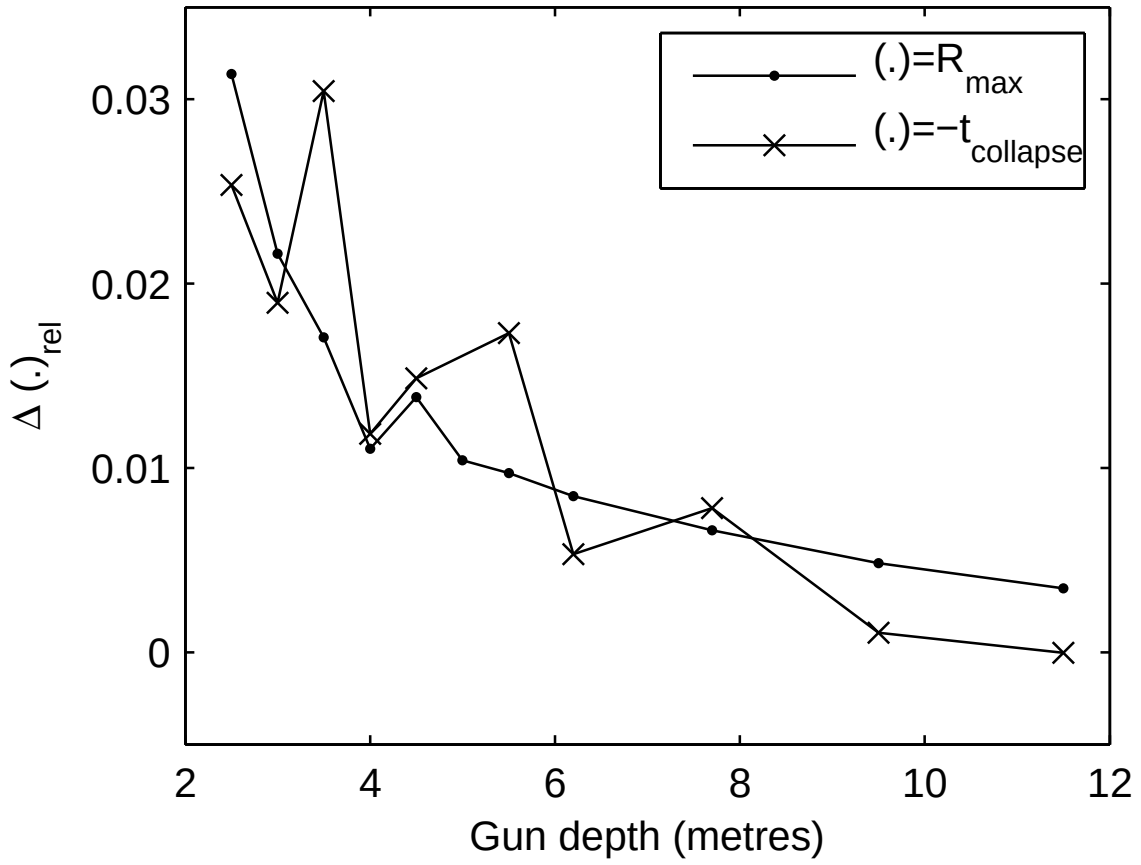
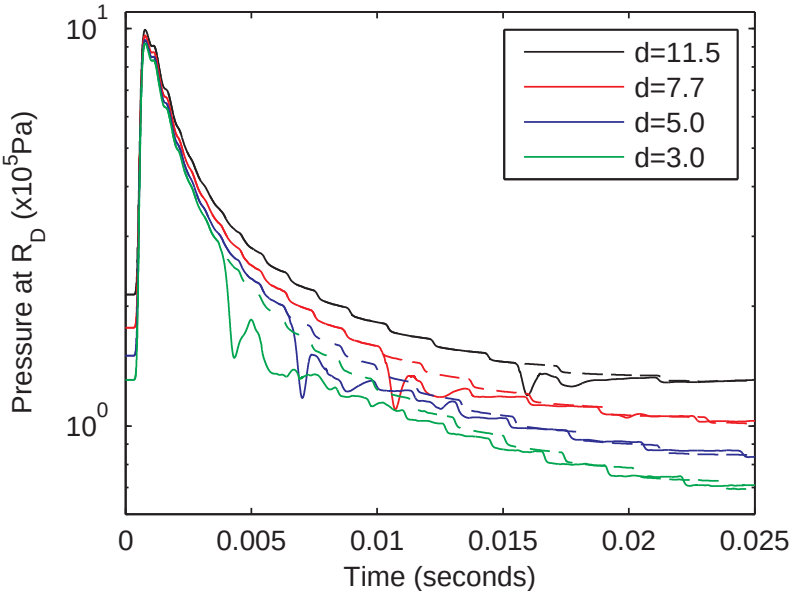
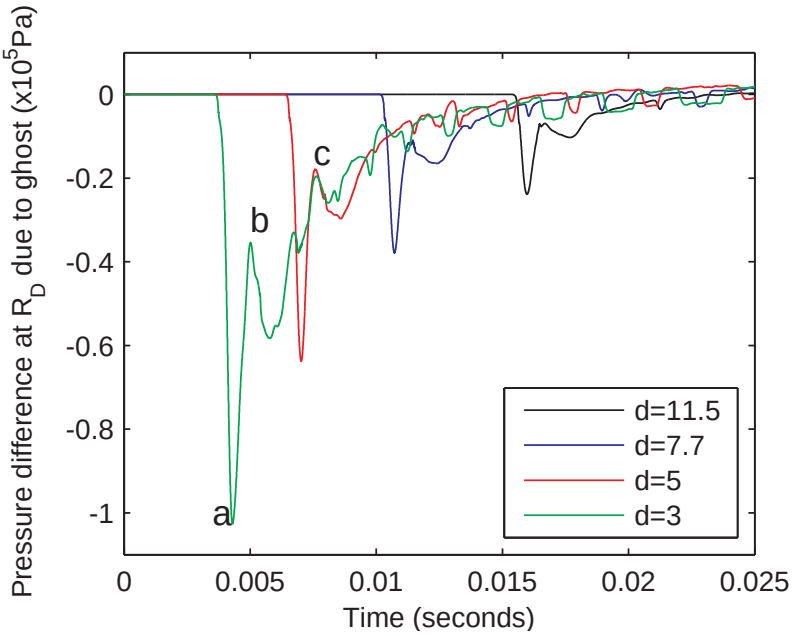


Figure 3: The influence of the ghost on the maximum bubble radius $R_{\max}$ and the bubble period $t_{collapse}$ as the depth of the gun is varied. Note that the sign of the trace for $t_{collapse}$ has been changed - the ghost actually causes a reduction in bubble period.

KING –



(a)



(b)

Figure 4: (a) - Variation of pressure on Γ at $\theta = \pi/2$ with time, without the ghost (dashed lines), and with the ghost (solid lines), for bubbles at depths of $d = 11.5$, 7.7, 5.0 and 3.0 metres. (b) - Variation of $\Delta p_{R_D,\pi/2}$ with time, for bubbles at depths of $d = 11.5$, 7.7, 5.0 and 3.0 metres. $\Delta p_{R_D,\pi/2} = p\left(R_D, \pi/2\right)_G - p\left(R_D, \pi/2\right)_{NG}$ is the difference between the pressure at $(R_D, \pi/2)$ with and without the ghost.

KING –

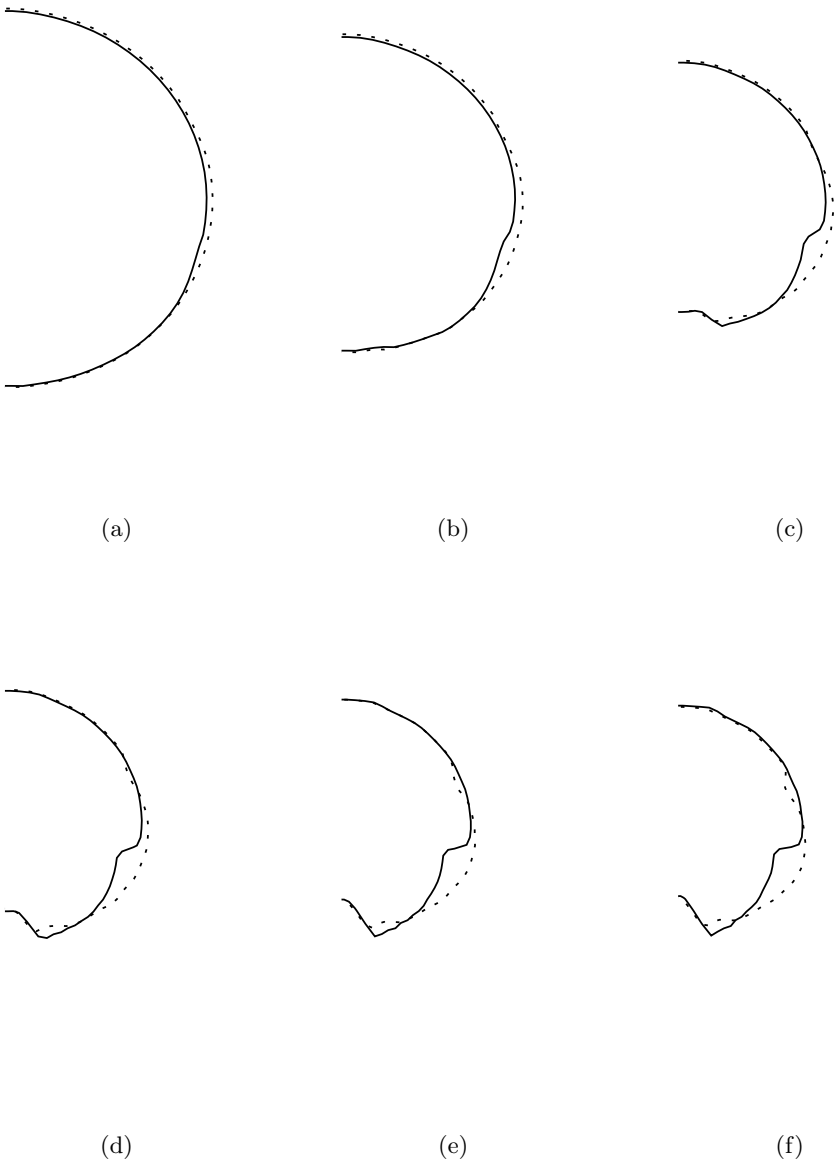


Figure 5: Bubble shapes at different times during collapse, with the ghost (dashed line) and without the ghost (solid line). $d = 7.7$ metres. (a) $t = 65.3$ms; (b) $t = 67.7$ms; (c) $t = 69.9$ms; (d) $t = 70.9$ms; (e) $t = 71.8$ms; (f) $t = 72.7$ms.

KING –

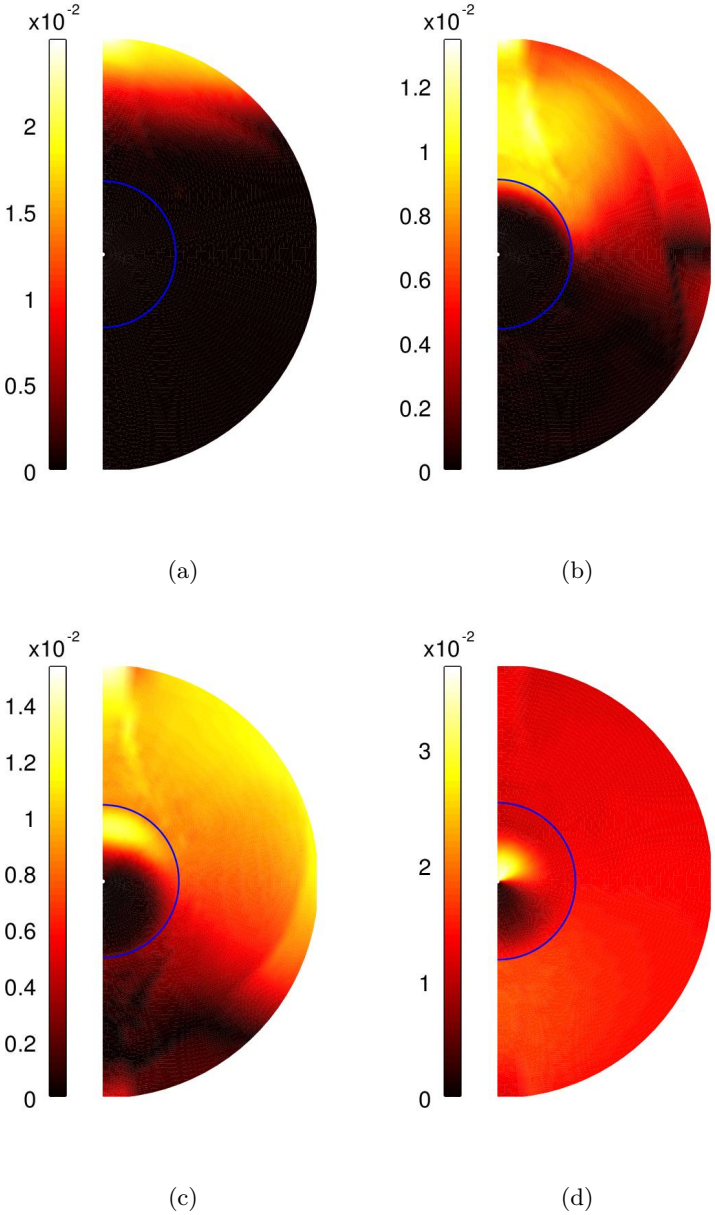


Figure 6: The ghost wave impacting on the bubble. The intensity of the image corresponds to the difference between the velocity magnitude with and without the ghost. The blue line represents the air-water interface. (a) $t = 10.21$ms; (b) $t = 10.81$ms; (c) $t = 11.41$ms; (d) $t = 12.31$ms.

KING –

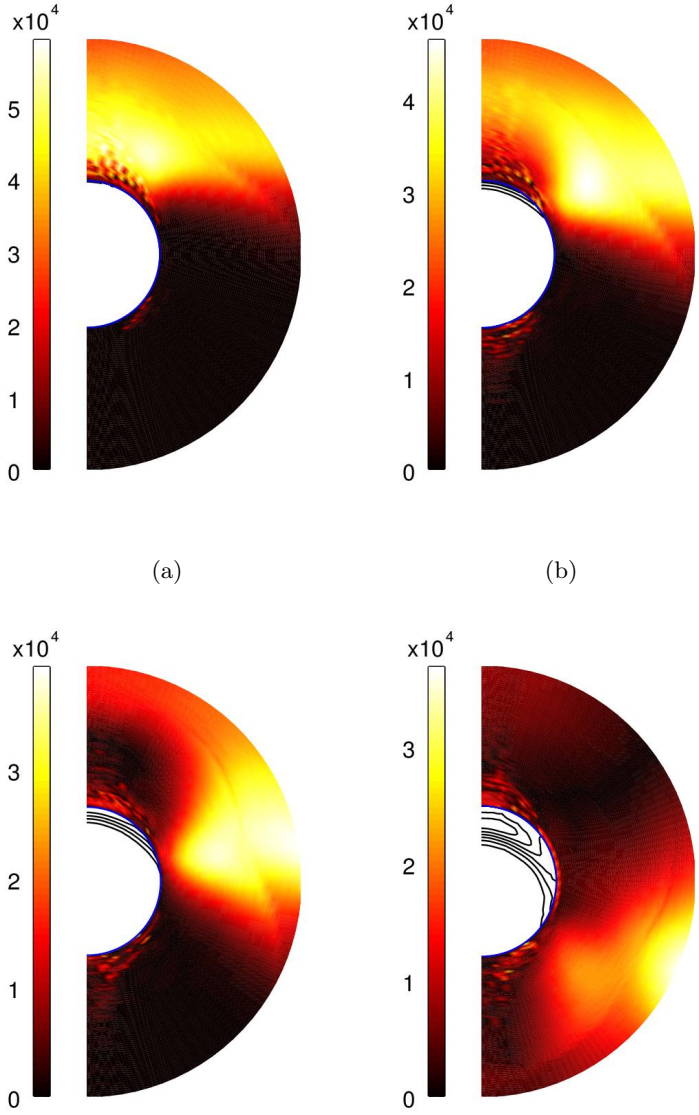
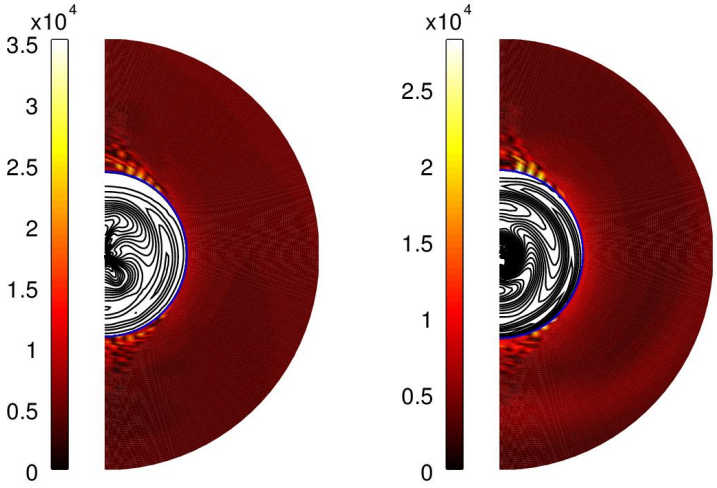


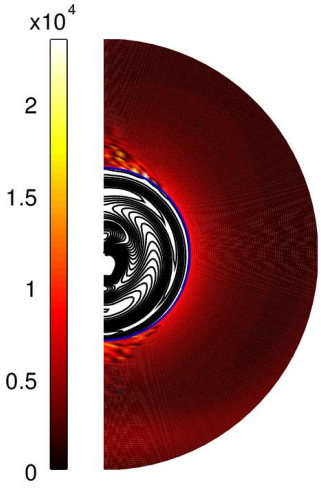
Figure 7: The ghost wave impacting on the bubble, modelled with $\delta r = 0.01$ metres. The blue line represents the air-water interface. Outside the bubble the intensity of the image corresponds to the magnitude of the difference in pressure with and without the ghost. The contours within the bubble show the difference in velocity magnitude with and without the ghost. (a) $t = 10.48$ms; (b) $t = 10.63$ms; (c) $t = 10.78$ms; (d)$t = 11.23$ms.

KING –



(a)

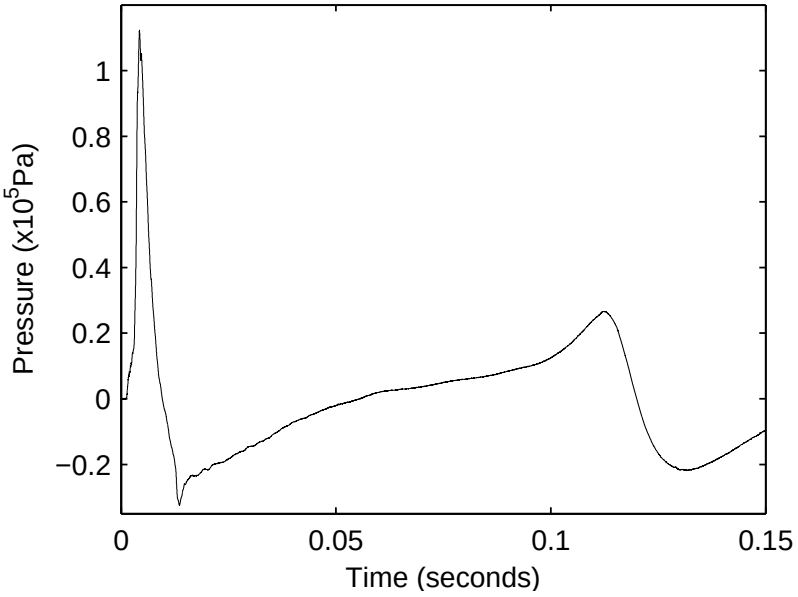
(b)



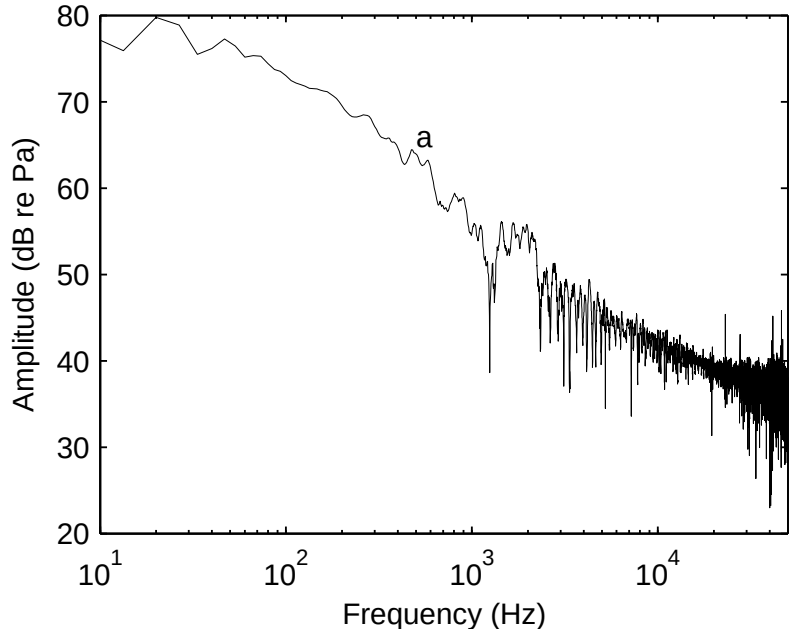
(c)

Figure 8: Plots showing the bubble after the ghost wave has impacted on the bubble, modelled with $\delta r = 0.01$ metres. The blue line represents the air-water interface. Outside the bubble the intensity of the image corresponds to the magnitude of the difference in pressure with and without the ghost. The contours within the bubble show the difference in velocity magnitude with and without the ghost. (a) $t = 14.53$ms; (b) $t = 15.43$ms; (c) $t = 16.78$ms.

KING –



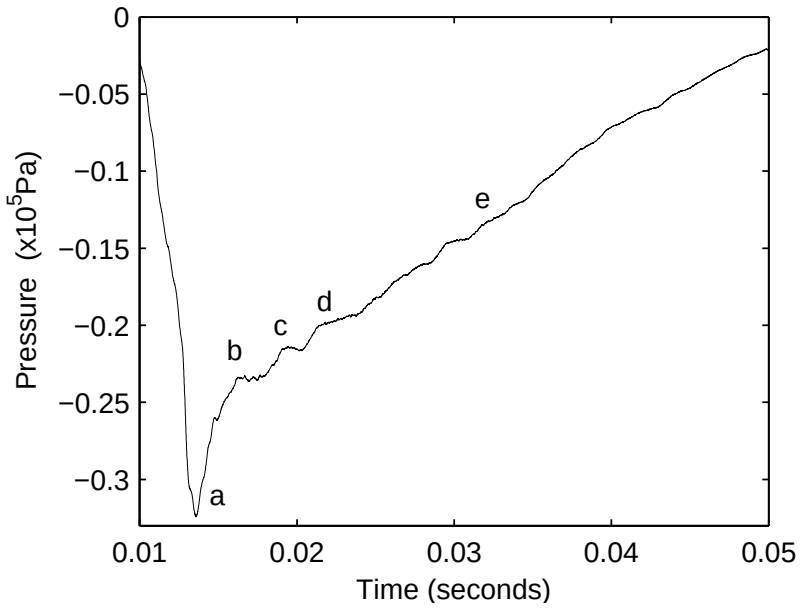
(a)



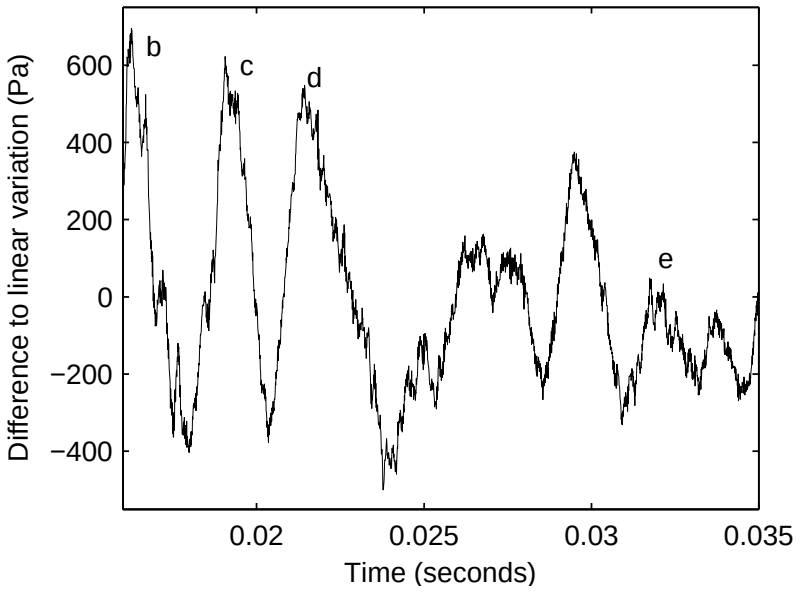
(b)

Figure 9: Near-field measurements from a 250cubic inch air gun at 6 metres depth. (a) pressure measurement; (b) amplitude spectrum.

KING –



(a)



(b)

Figure 10: The portion of the near-field measurement around the time the ghost wave impacts the bubble. (a) pressure measurement; (b) the difference between the direct measurement and a linear variation of pressure with time.

KING –